Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – KI141502

RANCANG BANGUN SISTEM *ENTERPRISE RESOURCE PLANNING* PADA MODUL *GENERAL LEDGER* BERORIENTASIKAN *MULTITENANCY* DENGAN MENGGUNAKAN SISTEM BASIS DATA TERDISTRIBUSI

Muhammad Ashari Adhitama
NRP 5112 100 133

Dosen Pembimbing
Sarwosri, S.Kom., M.T.
Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI141502

RANCANG BANGUN SISTEM *ENTERPRISE RESOURCE PLANNING* PADA MODUL *GENERAL LEDGER* BERORIENTASIKAN *MULTITENANCY* DENGAN MENGGUNAKAN SISTEM BASIS DATA TERDISTRIBUSI

Muhammad Ashari Adhitama
NRP 5112 100 133

Dosen Pembimbing
Sarwosri, S.Kom., M.T.
Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESIS – KI141502

DESIGN AND IMPLEMENTATION OF GENERAL LEDGER MODULE ON ENTERPRISE RESOURCE PLANNING WITH MULTITENANCY ORIENTATION USING DISTRIBUTED DATABASE

Muhammad Ashari Adhitama
NRP 5112 100 133

Supervisor
Sarwosri, S.Kom., M.T.
Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

Rancang Bangun Sistem *Enterprise Resource Planning* pada Modul *General Ledger* Berorientasikan *Multitenancy* dengan Menggunakan Sistem Basis Data Terdistribusi

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Manajemen Informasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
Muhammad Ashari Adhitama
NRP : 5112 100 133

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Sarwosri, S.Kom., M.T. .....
NIP: 19760809 200112 2 001 (pembimbing 1)

Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc, Ph.D. .....
NIP: 19590803 198601 1 001 (pembimbing 2)

SURABAYA
JUNI 2016

[Halaman ini sengaja dikosongkan]

Rancang Bangun Sistem *Enterprise Resource Planning* pada Modul *General Ledger* Berorientasikan *Multitenancy* dengan Menggunakan Sistem Basis Data Terdistribusi

Nama Mahasiswa	: Muhammad Ashari Adhitama
NRP	: 5112 100 133
Jurusan	: Teknik Informatika ITS
Dosen Pembimbing 1	: Sarwosri, S.Kom., M.T.
Dosen Pembimbing 2	: Prof. Drs.Ec. Ir. Riyanarto Sarno M.Sc., Ph.D.

ABSTRAK

Enterprise Resource Planning adalah sebuah sistem yang dirancang untuk mengelola sebuah perusahaan besar dalam mengkoordinasikan semua sumber daya, informasi, dan aktivitas yang diperlukan untuk proses bisnis lengkap. Salah satu modul yang krusial dan yang telah dikerjakan pada tugas akhir ini yaitu General Ledger. General Ledger adalah suatu proses penjurnalan seluruh transaksi keuangan yang ada di perusahaan, termasuk pelaporan keuangan serta penentuan akun-akun finansial terkait. Kesalahan pada laporan keuangan seperti kesalahan pada pembuatan jurnal, pencatatan yang tidak sesuai dengan kebenaran, dan lainnya adalah hal yang sering ditemui dan kecil kemungkinannya untuk menghindari hal tersebut, serta untuk menelusuri kesalahan dalam laporan tersebut akan sulit apabila dilakukan secara manual.

Dalam tugas akhir ini, dibuat sebuah sistem untuk memudahkan akuntan dalam menelusuri kesalahan dalam laporan keuangan dari perusahaan, dikarenakan tidak hanya adanya fitur search dalam journal tetapi juga dapat dilihat jurnal-jurnal terkait dalam periode tertentu. Sehingga apabila terdapat kesalahan pada laporan, dapat ditelusuri melalui periode dalam hari tertentu.

Hasil eksperimen menunjukkan bahwa sistem ERP yang dirancang menggunakan basis data terdistribusi dapat tetap berjalan dengan baik ketika salah satu server lain mati. Pada accounting, dengan adanya rasio-rasio analisis laporan keuangan, dapat membantu para investor ataupun manager dalam menentukan posisi kesehatan keuangan dari perusahaan tersebut.

Kata kunci: *General Ledger, Enterprise Resource Planning, Financial Statement, Journal, Periode, Chart of Accounts, Ratio Analysis.*

Design and Implementation of General Ledger Module on Enterprise Resource Planning with Multitenancy Orientation Using Distributed Database

Student Name	: Muhammad Ashari Adhitama
NRP	: 5112 100 133
Major	: Teknik Informatika ITS
Supervisor 1	: Sarwosri, S.Kom., M.T.
Supervisor 2	: Prof. Drs.Ec. Ir. Riyanarto Sarno M.Sc., Ph.D.

ABSTRACT

Enterprise Resource Planning is a system that is designed to manage an enterprise on coordinating all resources, information, and activities needed to fulfill business processes. One of the crucial modules and has been done in this final project is an general ledger module. General ledger is a journaling process of all financial transactions done by the company, including creating financial statements and determining financial accounts. Mistakes done in the financial statements like journal errors, recordings that doesn't based on truth, and the other mistakes are things that usually being faced and a chance to avoid the mistakes are small, besides that, to search for errors manually without a system is a work hard.

In this final project, a system is designed to facilitate accountant to search for errors on the enterprise's financial statements, not only there is a search feature on journal management, but also the accountant can check on which period the error is shown and then check which journal contains an error input.

The experimental results showed that ERP systems are designed using a distributed database can still run well when one of the other servers die. In accounting, with the ratios of financial

statement analysis, can help investors or managers in determining the position of the financial health of the company.

Keyword: *General Ledger, Enterprise Resource Planning, Financial Statement, Journal, Period, Chart of Accounts, Ratio Analysis.*

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas tkhir yang berjudul:

"*Rancang Bangun Sistem Enterprise Resource Planning pada Modul General Ledger Berorientasikan Multitenancy dengan Menggunakan Sistem Basis Data Terdistribusi*"

Melalui lembar ini, penulis hanya ingin menyampaikan ucapan terima kasih dan penghormatan yang sebesar-besarnya kepada:

1. Ayah, Ibu, Kakak, Adik dan seluruh kerabat keluarga yang selalu memberikan dukungan penuh untuk menyelesaikan tugas akhir ini.
2. Bapak Riyanarto Sarno beserta keluarga dan Ibu Sarwosri beserta keluarga selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberikan petunjuk selama proses pengerjaan tugas akhir ini.
3. Bapak dan ibu dosen Jurusan Teknik Informatika ITS yang telah banyak memberikan ilmu dan bimbingan bagi penulis.
4. Seluruh staf dan karyawan FTIf ITS yang banyak memberikan kelancaran administrasi akademik kepada penulis.
5. Al Aji, Akbar, Vicky, Arya, Nicko, Afina, Yusi, Melinda selaku teman ERP seperjuangan.
6. Teman-teman seperjuangan anak didik Tugas Akhir Prof. Riyanarto Sarno yaitu Pradipta, Baiquni, Aldrin, Luqman, Argyanto, Ardhya, Maya, Kelly, dan Andrean
7. Teman-teman angkatan 2012 jurusan Teknik Informatika ITS yang telah menemani perjuangan selama 4 tahun ini

atas saran, masukan, dan dukungan terhadap pengerjaan tugas akhir ini.

8. Rizka Azalea Nadiaputri yang tidak henti-hentinya selalu memberikan semangat dan membangkitkan *mood* penulis dari pagi hingga malam.
9. Oryza Sativa, S.E. yang bersedia meluangkan waktunya mengajari penulis berbagai macam hal terkait dengan akuntansi keuangan.
10. Serta pihak-pihak lain yang namanya tidak dapat penulis sebutkan satu -persatu.
11. Himpunan Mahasiswa Hormat Orang Tua yang selalu mengangkat semangat mental penulis dari pagi hingga tengah malam.

Bagaimanapun juga penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Juli 2016

Muhammad Ashari Adhitama

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

DAFTAR ISTILAH

Make to Stock
Proses bisnis yang dimulai dari menyetok barang.
Make to Order
Proses bisnis yang dimulai dari adanya pemesanan barang.
Chart of Accounts
Akun finansial untuk transaksi
Accounting Period
Periode akuntansi untuk penjurnalan
Financial Journal
Jurnal transaksi finansial maupun non-finansial yang disimpan oleh akuntan
Balance Sheet
Laporan neraca keseimbangan dari perusahaan
Income Statement
Laporan laba-rugi dari perusahaan
Cash Flow
Laporan aliran kas yang ada di perusahaan
Debit
Nilai yang bertambah pada suatu akun
Kredit
Nilai yang berkurang pada suatu akun
Parent
Node yang berada di atas node lain secara langsung
Node
Sebuah element dalam tree. Berisi informasi
Child
Cabang langsung dari sebuah node.
Sibling
Sebuah node lain yang memiliki parent yang sama.
ERP
Aplikasi yang digunakan untuk merencanakan dan mengelola sumber daya perusahaan meliputi dana, manusia, mesin, material dan kapasitas produksi yang berpengaruh luas mulai dari

manajemen paling atas hingga operasional di sebuah perusahaan agar dapat dimanfaatkan secara optimal.

Odoo

Salah satu aplikasi ERP yang bersifat *open source*.

InoERP

Salah satu aplikasi ERP yang bersifat *open source*.

Adempiere

Salah satu aplikasi ERP yang bersifat *open source*.

Simulasi bisnis

Permainan bisnis yang dirancang agar peserta mengalami dan berlatih menjalankan suatu perusahaan.

SQL

Bahasa yang dipergunakan untuk mengakses data dalam basis data.

INDEKS

S

W

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1 Latar Belakang

Laporan keuangan adalah salah satu dokumen vital yang dapat menentukan nasib suatu perusahaan, baik keuangan bulanan maupun tahunan. Laporan keuangan juga dapat digunakan untuk memperkirakan *sales* dan *demand* dari produk yang akan dijual, penentuan jumlah dari masing-masing barang yang akan diproduksi perusahaan ke depannya, serta menentukan keputusan yang akan diambil oleh *manager* dan *CEO* untuk pengembangan perusahaan. Kesalahan pada laporan keuangan seperti kesalahan pada pembuatan jurnal, pencatatan yang tidak sesuai kebenaran, dan lainnya adalah hal-hal yang sering ditemui dan kecil kemungkinannya untuk menghindari hal tersebut, serta untuk menelusuri kesalahan dalam laporan tersebut akan sulit apabila dilakukan secara manual, oleh karena itu dibutuhkan pengelola laporan yang berintegrasi dengan sebuah penyimpanan yang berisikan semua data-data keuangan dan transaksi yang terjadi.

Konsep *General Ledger* pada sistem *ERP (Enterprise Resource Planning)* adalah untuk meminimalisir kesalahan-kesalahan pada pembuatan laporan keuangan yang biasa terjadi. Modul ini dapat membantu para akuntan untuk menelusuri kesalahan pada jurnal yang dibuat (dan diterima), pencatatan peredaran uang, dan lainnya karena sudah terintegrasi dengan basis data sehingga kesalahan dapat dengan mudah ditelusuri dan ditemukan.

Pada sistem sebelumnya, telah dikembangkan sebuah domain *general ledger (accounting)* yang terfokus pada laporan-laporan seperti *balance sheet*, *income statement*, serta *handle*

jurnal, akun, serta periode yang telah terdata dengan baik. Sistem ini menggunakan arsitektur berorientasi *service* (SOA) dengan model pengembangan *Model-View- Controller* (MVC) dan *Workflow* untuk .NET, serta penggunaan database model *Separated Database*.

Dalam pengembangan sistem kali ini, pada modul *general ledger* akan didetilkan pada bagaimana akun dan jurnal dapat mengontrol laporan-laporan yang akan dibuat, dan pengimplementasian periode agar laporan yang dibuat bisa sesuai dengan periode yang ditentukan, serta pembuatan jurnal secara otomatis apabila ada transaksi yang dilakukan oleh modul lainnya. Sistem ini menggunakan model pengembangan *Model- View-Controller* (MVC) dan *workflow* untuk PHP, serta penggunaan database model *distributed database.*

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana aplikasi ERP modul *general ledger* dapat menggantikan proses akuntansi manual dan meningkatkan kinerja suatu perusahaan ?
2. Bagaimana penerapan algoritma *Tree* pada submodul *Chart of Accounts ?*
3. Bagaimana mengaplikasikan *multitenancy* pada basis data terdistribusi terkait modul *General Ledger* pada ERP?
4. Bagaimana menangani kegagalan sistem pada sebuah *database server* sehingga kinerja *tenant* lainnya tidak terganggu?

1.3 Batasan Permasalahan

Batasan Masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut :

1. ERP yang dibangun akan berorientasi *Multitenancy*
2. Terdapat beberapa sub-modul pada modul *general ledger :*
    a. *Financial Statement,* terdiri dari :
        1. *Balance Sheet*
        2. *Income Statement*
        3. *Cash Flow*
    b. *Period Management*
    c. *Journal Management*
    d. *Chart of Accounts,* terdiri dari :
        i. *Chart of Accounts*
        ii. *Account Balance*
3. Hasil dari tugas akhir ini adalah menghasilkan modul *general ledger* pada sistem *ERP* yang dibangun dengan arsitektur *distributed database.*
4. Algoritma yang diimplementasikan untuk pembuatan *Chart of Accounts* adalah Algoritma *Binary Tree*.

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Membuat aplikasi yang dapat mengontrol jurnal dan data-data transaksi finansial secara tepat dan akurat sesuai dengan periode akuntansi yang ditentukan, begitu juga dengan laporan keuangan yang akan dihasilkan.
2. Membuat aplikasi yang dapat memodelkan akuntansi sesuai standar ERP dan Akuntansi Keuangan.
3. Membuat aplikasi yang dapat mengimplementasikan *multitenancy* dan melakukan pengembangan terhadap sistem sebelumnya.
4. Menerapkan algoritma *Tree* pada submodul *Chart of Accounts.*

1.5 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

1. **Studi literatur**

   Pada tahap ini dilakukan untuk fokus pada pemrograman PHP serta mempelajari dasar dari akuntansi seperti pembuatan laporan keuangan serta memonitor jurnal dan akun, sehingga sistem yang dibangun dapat sesuai dengan kebutuhan akuntansi pada umumnya, serta pembelajaran proses bisnis yang akan dijalankan pada ERP yang dibangun.

2. **Analisis dan Perancangan Sistem**

   Pada tahap analisis dan desain, akan dilakukan perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang digunakan. Tahap ini mendefinisikan alur dari implementasi, perancangan database, perancangan *prototype* sistem. Desain perangkat lunak akan menggunakan sistem rancang MVC (*Model-View-Controller*) menggunakan framework Yii 2.0.

3. **Implementasi Perangkat Lunak**

   Pada tahap ini akan dilakukan implementasi dengan cara menerapkan algoritma-algoritma pada sistem yang dibangun menggunakan bahasa pemrograman PHP dengan Framework Yii 2.0, serta pengolahan database menggunakan MySQL.

4. **Pengujian dan evaluasi**

   Pada tahapan pengujian dan evaluasi, akan dilakukan uji coba dari sistem yang telah dibangun dan akan

dilakukan evaluasi dari segi fungsionalitas serta kesesuaian dengan rencana awal yang diajukan, dan pada tahap ini pula akan dicari *bug* yang dapat menimbulkan kesalahan sehingga dapat diperbaiki.

Untuk pengujian kemampuan *multitenant* yang menggunakan arsitektur *distributed database*, akan digunakan 5 komputer yang akan bertindak sebagai *tenant* dan untuk memraktikkan dengan cara mematikan salah satu dari ketiga komputer (*tenant*) tersebut dan memastikan apakah *tenant* lain masih dapat menjalankan proses bisnisnya tanpa terpengaruh dari *tenant* lainnya yang sudah dimatikan tersebut,

5. Penyusunan buku Tugas Akhir

Pada tahap ini dilakukan pendokumentasian dan pelaporan dari seluruh konsep, dasar teori, implementasi, proses yang telah dilakukan, dan hasil-hasil yang telah didapatkan selama pengerjaan tugas akhir.

1.6 Sistematika Penulisan

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

Bab II Dasar Teori

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan tugas akhir ini.

Bab III Metode Pemecahan Masalah

Bab ini membahas mengenai metode yang digunakan untuk memecahkan masalah yang dipaparkan pada rumusan permasalahan.

Bab IV Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur, proses dan perancangan antarmuka pada sistem.

Bab V Implementasi

Bab ini berisi implementasi dari perancangan perangkat lunak sistem dan implementasi fitur-fitur penunjang sistem.

Bab VI Pengujian dan Evaluasi

Bab ini membahas pengujian dengan metode pengujian subjektif untuk mengetahui penilaian aspek kegunaan (*usability*) dari perangkat lunak dan pengujian fungsionalitas yang dibuat dengan memperhatikan keluaran yang dihasilkan serta evaluasi terhadap fitur-fitur pada system

Bab VII Kesimpulan

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka
Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Daftar Istilah
Merupakan daftar istilah yang berhubungan dengan tugas akhir.

Indeks
Merupakan daftar list kata-kata penting pada buku tugas akhir.

Lampiran
Merupakan bab tambahan yang berisi tambahan-tambahan yang penting pada aplikasi ini

[Halaman ini sengaja dikosongkan]

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan Tugas Akhir. Teori-teori tersebut meliputi *Enterprise Resource Planning (ERP), Multitenancy, Distributed Database, Accounting,* proses bisnis ERP, *Financial Statement Analysis* dan *Accrual Method.*

2.1 Penelitian Terkait

Tugas Akhir ini merupakan pengembangan berkelanjutan dari riset atau penelitian tentang ERP. Pada Tugas Akhir yang sebelumnya, telah dibangun sebuah aplikasi bernama ERP 2013. Pada aplikasi ERP 2013, aplikasi dikembangkan dengan sistem *multitenancy* yaitu dalam satu sistem ERP, dapat digunakan untuk beberapa perusahaan dengan satu aplikasi yang sama sehingga lebih efektif dalam penggunaan memori komputer server. Sistem database yang digunakan adalah *separated database,* yaitu setiap data perusahaan disimpan di *database* yang berbeda. Dengan sistem *separated database* tersebut, aplikasi telah menghasilkan fungsionalitas yang optimal, tetapi masih kurang efektif dalam pengelolaan *database.* Hal itu dikarenakan ketika komputer server sedang tidak aktif atau mati, maka *database* tersebut tidak dapat diakses.

Pada tugas akhir ini, dikembangkan aplikasi ERP yang bernama ERP 2016, sistem dibangun dengan menggunakan sistem *distributed database* yaitu memisahkan *database* setiap *tenant* ke dalam komputer/*server* tersendiri yang dibedakan oleh IP *address* setiap komputer/*server.* Hal ini bertujuan untuk mencegah kegagalan sistem jika sebuah komputer/*server* mengalami *down.* Maka *database* masih bisa diakses melalui komputer server lain.

Modul pada ERP 2016 terdiri dari Modul *General Ledger*, *Account Payable* and *Account Receivable*, *Inventory and Warehouse Management, Production, Sales and Distribution, Finance, Purchasing, Asset Management* dan *Human Resource*.

Masing-masing modul memiliki alur keterkaitan dengan modul yang lain. Perbedaan modul yang terdapat pada ERP 2013 dan ERP 2016 ditunjukkan pada Tabel 2.1.

Tabel 2.1. Perbedaan ERP 2013 dan ERP 2016

No.	Nama Modul ERP 2013	Nama Modul ERP 2016
1	*Account Payable & Receivable*	*Account Payable & Receivable*
2	*General Ledger*	*General Ledger*
3	*Cost Accounting*	*Finance*
4	*Customer Relationship Management*	*Sales and Distribution*
5	*Production Planning*	*Production*
6	*Supplier Relationship Management*	*Purchasing*
7	*Inventory*	*Inventory and Warehouse Management*
8	*Assets Management*	*Assets Management*
9	-	*Human Resource Management*

Kemudian pada modul *General* Ledger, pada aplikasi ERP 2013 terdapat 6 sub-modul yaitu: *Chart of accounts, Period, Journals, Balance Sheet, Income Statement,* serta *Cash Flow.* Ternyata submodul-submodul tersebut masih belum terbilang lengkap biarpun sudah dapat memenuhi kebutuhan *General Ledger* secara umum. Oleh karena itu pada aplikasi ERP 2015 ditambahkan sebuah submodul yang bernama *Account Balance*.

Tabel 2.2. Perbedaan ERP 2013 dan ERP 2016

No	ERP 2013	ERP 2016	Keterangan
1	*Journal*	*Journal*	Memiliki fungsi yang sama untuk menangani jurnal transaksi-transaksi yang terjadi di perusahaan. Hanya saja pada ERP 2016, penjurnalan bisa dibuat

			secara otomatis dan dapat diduplikasi.
2	*Balance Sheet*	*Balance Sheet*	Memiliki fungsi yang sama untuk menangani laporan neraca keseimbangan. Namun pada ERP 2016, laporan yang dicetak tidak hanya berupa harian, namun bisa tahunan, bulanan, harian dan 3 bulanan.
3	*Income Statement*	*Income Statement*	Memiliki fungsi yang sama untuk menangani laporan laba rugi. Namun pada ERP 2016, laporan yang dicetak tidak hanya berupa harian, namun bisa tahunan, bulanan, harian, dan 3 bulanan.
4	*Cash Flow*	*Cash Flow*	Memiliki fungsi yang sama untuk menangani laporan aliran kas. Namun pada ERP 2016, laporan yang dicetak tidak hanya berupa harian, namun bisa tahunan, bulanan, dan harian.
5	-	*Account Balance*	Memiliki fungsi untuk menampilkan balance dari masing-masing akun, serta jumlah debit dan kreditnya dari awal perusahaan dibangun sampai terakhir transaksi.

Pada modul *Accounting* ERP 2013, tidak terdapat pembahasan analisis laporan keuangan serta sulitnya untuk mencetak laporan yang hanya bisa dilakukan per hari, serta sulit untuk mengetahui jumlah uang yang ada sekarang serta profit perkembangan per hari, karena tidak terdapat grafis perkembangan keuntungan dan tidak ada *account balance* yang dapat memudahkan untuk melacak *balance* pada masing-masing akun. Pada pengembangan ERP kali ini, akan dikembangkan

sebuah ERP bernama ERP 2016. ERP 2016 memiliki modul yang juga dapat menangani pelaporan keuangan, yaitu modul *accounting* yang lebih terfokus ke *financial accounting*. Pada modul *accounting* pada ERP 2016 telah mampu menangani beberapa kekurangan yang terdapat pada modul *accounting* pada ERP 2013. Untuk masalah analisis laporan keuangan, ERP 2016 memiliki analisis terkait laporan keuangan dalam bentuk grafis sehingga memudahkan bagi *manager* untuk menganalisis laporan keuangan yang ada. Untuk masalah pelaporan keuangan, ERP 2016 memiliki laporan keuangan seperti *balance sheet*, *income statement*, serta *cash flow* yang dapat dilihat berdasarkan harian, bulanan, bahkan tahunan.

2.2 ERP

ERP / Perencanaan Sumber Daya Perusahaan adalah sistem informasi yang diperuntukkan bagi perusahaan manufakturing maupun jasa yang berperan mengintegrasikan dan mengotomasikan proses bisnis yang berhubungan dengan aspek operasi, produksi maupun distribusi diperusahaan tersebut [1]. Jadi ERP adalah sebuah termologi yang diberikan kepada sistem informasi yang mendukung transaksi atau operasi sehari-haridalam pengelolaan sumber daya perusahaan meliputi dana, manusia, mesin, suku cadang, waktu, material dan kapasistas.

Keuntungan penggunaan ERP diantaranya adalah Integrasi data keuangan, standarisaasi proses operasi, standarisasi data dan informasi, penurunan inventori dan tenaga kerja, peningkatan servis level dan kontrol keuangan dan penurunan waktu yang dibutuhkan untuk mendapatkan informasi.

Ada pun departemen-departemen atau bagian-bagian yang pada umumnya terintegrasi meliputi:

1. Pengelolaan keuangan (*Finance Resource Management*) dalam aplikasi ERP 2016 ini diwakili oleh modul *Account Payable*, *Account Receivable*, *Fixed Asset*, *Cost Management*, *General Ledger*
2. Pengelolaan rantai pemasokan (*Supply Chain Management*) diwakili oleh modul *Purchasing*

3. Perencanaan produksi dan manufaktur (*Manufacturing Resource Planning*) diwakili oleh modul *Production* dan *Inventory*
4. Pengelolaan sumber daya manusia (*Human Resource Management*) diwakili oleh modul *Human Resource.*
5. Pengelolaan relasi dengan pelanggan (*Customer Relationship Management*) diwakili oleh modul *Sales Order*.

Departemen-departemen tersebut diintegrasikan melalui suatu sistem ERP terpusat, seperti yang ditunjukkan dalam Gambar 2.1

Gambar 2.1 Diagram integrasi ERP secara umum

Pada tugas akhir ini, akan dibangun satu modul dari domain fungsi yang termasuk dalam sistem ERP yaitu *General Ledger*. Penjelasan lebih mendetail lagi mengenai modul terkait adalah sebagai berikut

2.2.1 Accounting (General Ledger)

Accounting (General Ledger) atau akuntansi adalah pengukuran, penjabaran, atau pemberian kepastian mengenai informasi yang akan membantu manajer, investor, otoritas pajak dan pembuat keputusan lain untuk membuat alokasi sumber daya keputusan di dalam perusahaan, organisasi, dan lembaga pemerintah [2]. Akuntansi adalah seni dalam mengukur, berkomunikasi dan menginterpretasikan aktivitas keuangan [3]. Secara luas, akuntansi juga dikenal sebagai "bahasa bisnis". Akuntansi bertujuan untuk menyiapkan suatu laporan keuangan yang akurat agar dapat dimanfaatkan oleh para manajer, pengambil kebijakan, dan pihak berkepentingan lainnya, seperti pemegang saham, kreditur, atau pemilik [4]. Pencatatan harian yang terlibat dalam proses ini dikenal dengan istilah pembukuan. Akuntansi keuangan adalah suatu cabang dari akuntansi dimana informasi keuangan pada suatu bisnis dicatat, diklasifikasi, diringkas, diinterpretasikan, dan dikomunikasikan.

Accounting pada ERP bertugas untuk mengelola seluruh laporan keuangan, jurnal dari transaksi, serta mengelola periode (buka-tutup buku) [5]. Tujuan utama dari *accounting* adalah untuk menyetujui jurnal yang diterima dari transaksi yang telah dilakukan oleh departemen lainnya, lalu menjadikan jurnal-jurnal yang telah disetujui tersebut sebuah laporan keuangan seperti *balance sheet*, *income statement*, maupun *cash flow*, serta pembuatan dan penyusunan akun-akun terkait sehingga mudah untuk mengelompokannya dan menjadikannya laporan keuangan.

2.2.1.1 Chart of Accounts

Chart of Accounts atau yang di dalam bahasa Indonesia disebut Bagan Akun, adalah satu daftar rangkaian akun-akun yang sudah dibuat atau disusun secara sistematis dan teratur dengan menggunakan simbol-simbol huruf, angka, atau paduan antara keduanya yang bermanfaat untuk membantu pemrosesan data, baik secara manual maupun terkomputerisasi, agar lebih mudah diproses, dikontrol, dan dilaporkan.

2.2.1.2 Period Management

Period adalah waktu dimana sebuah transaksi dimulai dan dihentikan. Period dimulai apabila ada jurnal yang masuk ke sistem dan dihentikan apabila sudah berganti hari. *Period* dalam modul ini digunakan hanya untuk periode akuntansi, tidak berpengaruh terhadap modul lainnya.

2.2.1.3 Journal Management

Journal adalah rekaman transaksi yang telah dijalankan pada perusahaan, bisa berupa penjualan, produksi, pembelian, pengiriman barang dan pembayaran untuk pengeluaran lainnya. Dalam *journal* terdapat rincian dari masing-masing transaksi yang telah terjadi. *Journal* memerlukan 2 (dua) sub-modul lainnya yaitu: *chart of accounts* dan *period*.

2.2.1.4 Financial Statements

Financial Statements atau yang biasa disebut laporan keuangan adalah rekapan dari seluruh jurnal yang telah masuk ke sistem. *Financial statements* terbentuk dari jurnal-jurnal yang telah masuk ke dalam sistem, baik debit maupun kreditnya. Ada 3 tipe *financial statements*:

2.2.1.4.1 Balance Sheet

Balance Sheet atau neraca keseimbangan adalah salah satu jenis laporan keuangan. Neraca keseimbangan menggambarkan bagaimana proses tingkat kesehatan finansial yang ada di perusahaan.

2.2.1.4.2 Income Statement

Income Statement atau laporan pendapatan adalah jenis laporan keuangan yang memperlihatkan pengeluaran dan pendapatan uang dari transaksi pembelian, penjualan maupun pembayaran listrik, air dan lainnya.

2.2.1.4.3 Cash Flow

Cash Flow atau aliran kas adalah jenis laporan keuangan yang menampilkan arus uang yang terjadi di perusahaan. Untuk mendapatkan hasil pada cash flow bisa dilihat pada jurnal-jurnal yang menggunakan cash.

2.3 Multitenancy

Multitenancy merupakan suatu prinsip dari arsitektur perangkat lunak, dimana sebuah perangkat lunak yang berjalan di atas *server* melayani banyak pengguna/*tenant*. Dengan prinsip *multitenant* ini, sebuah perangkat lunak dirancang untuk memiliki partisi data yang berbeda dan dapat dikonfigurasi.

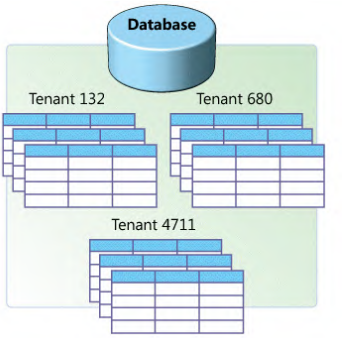
Arsitektur data *multitenancy* digunakan karena dinilai memadai dan cukup aman dalam mengatasi masalah kurang kepercayaan *tenant* untuk menyerahkan kontrol data bisnis *tenant* kepada pihak ketiga. Terdapat 3 jenis *multitenancy*, antara lain:

1. ***Separated Database*** yaitu data setiap *tenant* disimpan pada *database* yang terpisah dengan *tenant* lain. Keuntungan arsitektur ini adalah mudah untuk mengatur kembali model data aplikasi yang digunakan. Tetapi memerlukan biaya yang cukup tinggi untuk menjaga peralatan *server* dan juga *back up* data dari setiap *tenant*. Model arsitektur *separated database* ditunjukkan pada Gambar 2.2.

Gambar 2.2. *Separated Database*

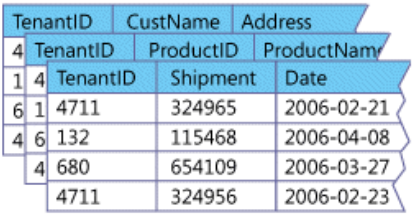
2. ***Shared Database, Separate Schema*** yaitu data setiap *tenant* disimpan pada satu *database* tetapi pada *schema* yang terpisah

dengan *tenant* lain. Keuntungan dari arsitektur ini adalah mudah untuk digunakan karena tabel yang dibentuk pada awalnya merupakan tabel standar, dan selanjutnya dapat diubah sesuai keinginan *tenant*. Akan tetapi apabila terjadi kegagalan maka perlu dilakukan perbaikan untuk semua tabel yang ada dalam *database.* Model arsitektur *shared database, separate schema* ditunjukkan pada Gambar 2.3.



Gambar 2.3. *Shared Database, Separate Schema*

3. ***Shared Database, Shared Schema*** yaitu data setiap *tenant* disimpan pada satu *database* dan satu *schema*. Keuntungan dari arsitektur ini adalah tidak memerlukan biaya yang tinggi, akan tetapi apabila terjadi kegagalan maka perlu dilakukan perbaikan untuk semua tabel yang ada dalam *database.* Model arsitektur *Shared Database*, *Shared Schema* ditunjukkan pada Gambar 2.4.



Gambar 2.4. *Shared database, Shared Schema*

2.4 Distributed Database

Basis data terdistribusi (BDT) adalah suatu basis data yang memiliki kontrol terpusat pada *server* manejemen dengan distribusi penyimpanan data yang tersebar di beberapa *server* data. BDT diterapkan dalam rangka untuk menjaga prinsip *high availability* pada aplikasi ERP. BDT memungkinkan aplikasi tetap berjalan jika terdapat sebuah *server* basis data mengalami *system failure*. Terdapat 2 mekanisme utama dalam BDT, yaitu replikasi dan fragmentasi.

2.4.1 Replikasi

Replikasi adalah mekanisme penyalinan seluruh atau sebagian *table* basis data ke beberapa *server data*. Setiap transaksi (penambahan, penghapusan, atau perubahan data) akan dieksekusi pada semua *server* data yang menyusun sistem BDT. Hal ini menyebabkan mekanisme replikasi membebani performa sistem, namun memiliki tingkat kompleksitas yang paling sederhana.

2.4.2 Fragmentasi

Fragmentasi adalah mekanisme penyalinan sebagian data atau struktur dari setiap tabel basis data. Transaksi basis data yang terjadi harus diolah terlebih dahulu oleh *server* manajemen untuk menentukan letak *server* data dari tabel yang akan terpengaruh. Hal ini menyebabkan mekanisme fragmentasi memiliki tingkat kompleksitas yang tinggi, namun memiliki performa yang lebih baik dibandingkan dengan mekanisme replikasi.

.

2.5 Proses Bisnis *ERP*

Sebelum menggunakan aplikasi ERP, maka pengguna harus memahami proses bisnis dalam satu perusahaan. Hal ini harus dikuasai karena aliran aplikasi ERP mengikuti proses bisnis suatu perusahaan. Ada beberapa proses bisnis perusahaan yang harus dianalisis yaitu pengadaan (*procurement*), produksi (*production*), penjualan (*sales/fullfillment*).

2.5.1 Proses Bisnis *Procurement*

Proses bisnis pengadaan merupakan proses dimana suatu perusahaan membeli barang untuk diproduksi. Proses bisnis ini merupakan awal dalam proses *Make to Stock*, sebuah proses dimana perusahaan membuat barang sebelum adanya pembelian. Proses bisnis ini dimulai dengan perusahaan membuat dokumen *purchase requisition* (PR). Kemudian dokumen PR ini setelah disetujui akan menjadi dokumen *purchase order* (PO) atau perintah pembelian barang. Setelah barang sudah dibeli, maka muncul dokumen *receive material* (RM) yang menandakan barang sudah diterima. Kemudian perusahaan mendapatkan tagihan atau *invoice* untuk dibayar. Terakhir perusahaan membayarkan sejumlah uang untuk melunasi tagihan tersebut. Terdapat gambar 2.5 untuk menjelaskan dan menggambarkan proses bisnis *procurement*.

Gambar 2.5. Proses Bisnis *Procurement*

2.5.2 Proses Bisnis Produksi

Proses bisnis produksi merupakan proses perusahaan memproduksi barang. Untuk membuat suatu barang, perusahaan harus mengikuti proses bisnis ini. Proses business ini dimulai dengan perusahaan membuat dokumen *request production*. Setelah produksi disetujui maka akan muncul dokumen *authorized production*. Produksi akan dapat berjalan ketika bahan mentah sudah tersedia dan dapat dikeluarkan dari gudang. Proses produksi dimulai ketika dokumen

create product sudah muncul. Setelah barang sudah jadi maka, barang akan dipindahkan dari tempat produksi ke gudang dengan adanya dokumen *receive finished goods*. Terdapat gambar 2.6 untuk menjelaskan dan menggambarkan proses bisnis produksi.

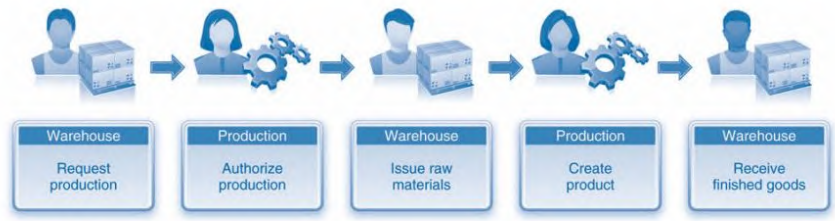


Figure 1-5: A production process

Gambar 2.6. Proses Bisnis *Production*

2.5.3 Proses Bisnis Penjualan

Figure 1-6: A fulfillment process

Gambar 2.7. Proses Bisnis Penjualan

Proses bisnis penjualan merupakan proses perusahaan menjual barang kepada pelanggan. Proses ini merupakan proses terpenting dalam simulasi bisnis. Proses bisnis ini dimulai dengan perusahaan membuat dokumen *sales order* untuk mengelola siapa yang membeli barang, barang yang dibeli, jumlah barang, harga barang, dan total harga. Kemudian proses dilanjutkan dengan mempersiapkan pengiriman barang. Barang akan dikirim dan muncul dokumen *shipment*. Setelah barang diterima oleh *customer*, maka perusahaan

mengirimkan tagihan atau *invoice* kepada pelanggan. Proses bisnis ini berakhir ketika pelanggan sudah membayar tagihan tersebut. Terdapat gambar 2.7 untuk menjelaskan dan menggambarkan proses bisnis penjualan.

2.6 Financial Statement Analysis

Analisis laporan keuangan atau *financial statement analysis* adalah merupakan suatu alat analisa yang digunakan oleh perusahaan untuk menilai kinerja keuangan berdasarkan data perbandingan masing-masing pos yang terdapat di laporan keuangan seperti laporan neraca, laba-rugi, dan arus kas dalam periode tertentu. Ada beberapa metode untuk menganalisis sebuah laporan keuangan dan dapat dituangkan dalam bentuk rasio, metode tersebut yaitu :

2.6.1 Current Ratio

Current ratio (rasio lancar) adalah rasio yang sangat berguna untuk mengukur kemampuan perusahaan dalam melunasi kewajiban-kewajiban jangka pendeknya, dimana dapat diketahui sampai seberapa jauh sebenarnya jumlah aktiva lancar perusahaan dapat menjamin utang lancarnya.

Rumus untuk menghitung *current ratio* adalah :

$$\frac{Current\ Assets}{Current\ Liabilities} \tag{2.1}$$

Terdapat 2 aspek dalam penghitungan *current ratio* seperti yang ditampilkan pada persamaan 2.1, yaitu *current assets* dan *current liabilities*. *Current Assets* adalah total aset yang dimiliki saat ini secara keseluruhan yang merupakan aktiva lancar, sedangkan *current liabilities* adalah total kewajiban yang harus dibayarkan (hutang jangka pendek) oleh perusahaan.

2.6.2 Quick Ratio

Quick Ratio (Rasio cepat) adalah sebuah rasio yang digunakan untuk mengukur kemampuan suatu perusahaan dalam menggunakan aktiva lancar untuk menutupi utang lancarnya.

Rumus untuk menghitung *quick ratio* adalah *:*

$$\frac{Quick\ Assets}{Current\ Liabilities} \tag{2.2}$$

Terdapat 2 aspek dalam penghitungan *quick ratio* seperti yang ditampilkan pada persamaan 2.2, yaitu *quick assets* dan *current liabilities. Quick Assets* adalah total aset yang dimiliki saat ini namun hanya berupa aktiva lancar seperti uang dan kas, sedangkan *current liabilities* adalah total kewajiban yang harus dibayarkan (hutang jangka pendek) oleh perusahaan.

2.6.3 Net Working Capital Ratio

Net Working Capital Ratio digunakan untuk mengukur likuiditas dari total aktiva perusahaan dan posisis modal kerja.

Rumus untuk menghitung *net working capital ratio* adalah :

$$\frac{Current\ Assets - Current\ Liabilities}{Total\ Assets} \tag{2.3}$$

Terdapat 3 aspek dalam penghitungan *net working capital ratio* seperti yang ditampilkan pada persamaan 2.3, yaitu *Current Assets*, *Current Liabilities*¸ dan *Total Assets*. *Current Assets* adalah total aset yang dimiliki saat ini secara keseluruhan yang merupakan aktiva lancar, *Current Liabilities* adalah total kewajiban yang harus dibayarkan (hutang jangka pendek) oleh perusahaan, dan *Total Assets* adalah total aset secara keseluruhan baik aset lancar maupun aset tetap.

2.6.4 Assets Turnover Ratio

Asset turnover ratio (ATO) atau disebut juga rasio perputaran total aktiva merupakan rasio yang mengukur tingkat efisiensi dan efektivitas dari perputaran maupun pemanfaatan total aktiva dalam menghasilkan penjualan.

Rumus untuk menghitung *assets turnover ratio* adalah :

$$\frac{Sales}{Average\ Total\ Assets} \tag{2.4}$$

Terdapat 2 aspek dalam penghitungan *assets turnover ratio* seperti yang ditampilkan pada persamaan 2.4, yaitu *sales* dan *average total assets. Sales* adalah total penjualan bersih yang diterima oleh perusahaan (tidak termasuk pajak, diskon, dan sebagainya), sedangkan *average total assets* adalah rata-rata total aset yang dimiliki perusahaan sebelum periode dimulai dan pada akhir periode.

2.6.5 Account Receivable Turnover Ratio

Account Receivable Turnover (Perputaran piutang) adalah suatu angka yang menunjukkan berapa kali suatu perusahaan melakukan tagihan atas piutangnya pada periode tertentu.

Rumus untuk menghitung *account receivable turnover ratio* adalah :

$$\frac{Sales}{Average\ Account\ Receivable} \tag{2.5}$$

Terdapat 2 aspek dalam penghitungan *account receivable turnover ratio* seperti yang ditampilkan pada persamaan 2.5, yaitu *sales* dan *average account receivable*. *Sales* adalah total penjualan bersih yang diterima oleh perusahaan (tidak termasuk pajak, diskon, dan sebagainya), sedangkan *average account receivable* adalah rata-rata hutang perusahaan kepada *supplier* sebelum periode dimulai dan pada akhir periode.

2.6.6 Return on Assets

Return on Assets (ROA) adalah salah satu bentuk dari rasio profitabilitas untuk mengukur kemampuan perusahaan dalam menghasilkan laba dengan menggunakan total aktiva yang ada dan setelah biaya-biaya modal (biaya yang digunakan mendanai aktiva) dikeluarkan dari analisis.

Rumus untuk menghitung *return on assets* adalah :

$$\frac{Net\ Income}{Average\ Total\ Assets} \tag{2.6}$$

Terdapat 2 aspek dalam penghitungan *return on assets* seperti yang ditampilkan pada persamaan 2.6, yaitu *net income* dan *average total assets*. *Net Income* adalah total keuntungan bersih yang didapat oleh perusahaan, sedangkan *average total assets* adalah rata-rata total aset yang dimiliki perusahaan sebelum periode dimulai dan pada akhir periode.

2.6.7 Profit Margin

Profit Margin digunakan untuk mengukur kemampuan perusahaan dalam mendapatkan laba bruto per penjualan.

Rumus untuk menghitung *profit margin* adalah :

$$\frac{Net\ Income}{Sales} \tag{2.7}$$

Terdapat 2 aspek dalam penghitungan *profit margin* seperti yang ditampilkan pada persamaan 2.7, yaitu *net income* dan *sales*. *Net Income* adalah total keuntungan bersih yang didapat oleh perusahaan, sedangkan *sales* adalah total penjualan bersih yang diterima oleh persahaan (tidak termasuk pajak, diskon, dan sebagainya).

2.7 PHP (PHP : *Hypertext Processor*)

PHP adalah singkatan dari "PHP: *Hypertext Prepocessor*", yaitu bahasa pemrograman yang digunakan secara luas untuk penanganan pembuatan dan pengembangan sebuah situs web dan bisa digunakan bersamaan dengan HTML. PHP diciptakan oleh Rasmus Lerdorf pertama kali tahun 1994. Pada awalnya PHP adalah singkatan dari "*Personal Home Page Tools*". Selanjutnya diganti menjadi FI ("*Forms Interpreter*"). Sejak versi 3.0, nama bahasa ini diubah menjadi "*PHP: Hypertext Prepocessor*" dengan singkatannya "PHP". PHP versi terbaru adalah versi ke-5. Berdasarkan survey Netcraft pada bulan Desember 1999, lebih dari sejuta *site* menggunakan PHP, diantaranya adalah NASA, Mitsubishi, dan RedHat.

2.8 Framework Yii 2.0

Yii adalah *framework* (kerangka kerja) PHP berbasis komponen, berkinerja tinggi untuk pengembangan aplikasi *web* berskala besar. Yii menyediakan *reusability* maksimum dalam pemrograman *web* dan mampu meningkatkan kecepatan pengembangan secara signifikan. Nama Yii (dieja sebagai /i:/) merupakan singkatan dari "**Yes It Is!**".

Yii adalah *framework* pemrograman umum Web yang bisa dipakai untuk mengambangkan semua jenis aplikasi *web*. Dikarenakan sangat ringan dan dilengkapi dengan mekanisme *caching* yang canggih. Yii sangat cocok untuk pengembangan aplikasi dengan lalu lintas tinggi seperti portal, forum, sistem manajemen konten (CMS), sistem *e-commerce*, dan lainnya.

Seperti kebanyakan PHP *framework*, Yii adalah *framework* yang berbasis MVC. Yii dapat melampaui *framework* lainnya dalam hal efisiensi, kekayaan fitur, dan kejelasan dokumentasi. Yii didesain dengan hati-hati dari awal agar sesuai untuk pengembangan aplikasi *web* secara serius. Yii bukan berasal dari produk pada beberapa proyek maupun konglomerasi pekerjaan pihak ketiga. Yii adalah hasil dari pengalaman kaya para pembuat pada pengembangan aplikasi *web* dan

investigasi *framework* pemrograman *web* dan aplikasi yang paling populer.

2.9 MySQL

MySQL adalah sebuah implementasi dari sistem manajemen basisdata relasional (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basis data yang telah ada sebelumnya; SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

2.10 Database Cluster

Database clustering adalah kumpulan dari beberapa *server* yang berdiri sendiri yang kemudian bekerjasama sebagai suatu sistem tunggal. Saat ini aplikasi *database* semakin berkembang, baik dalam hal kegunaan, ukuran, maupun kompleksitas. Hal ini secara langsung berdampak pada server *database* sebagai penyedia layanan terhadap akses *database*, konsekuensi dari semua itu adalah beban *database server* akan semakin bertambah berat dan mengakibatkan kurang optimalnya kinerja dari *server* tersebut.

Oleh karena itu diperlukan perancangan yang tepat dan handal dalam membangun *database* server. *Database* pada masa sekarang ini dituntut agar dapat berjalan dengan cepat, mempunyai kehandalan dan ketersediaan yang tinggi, dengan *clustering database* yang disimpan dapat terbagi ke beberapa mesin dan pada saat aplikasi berjalan, semua mesin yang menyimpan data tersebut dianggap sebagai satu kesatuan. Metode clustering seperti ini sangat baik untuk *load balancing* dan penanganan *system failure* karena kemampuan tiap mesin akan digunakan dan jika ada salah satu mesin yang mengalami *failure* maka

sistem tidak akan langsung terganggu karena mesin lain akan tetap berfungsi. Kemampuan *clustering* memungkinkan sebuah *database* tetap hidup dalam waktu yang lama.

2.11 MySQL Cluster

MySQL Cluster merupakan sebuah tipe basis data *(database*) yang dapat beroperasi dalam ukuran data yang relatif besar (maksimal dalam skala beberapa ratus *gigabyte*). MySQL Cluster adalah sebuah teknologi baru untuk memungkinkan *clustering* di dalam *memory database* dalam sebuah sistem *share-nothing*. Arsitektur *share-nothing* mengijinkan sistem dapat bekerja dengan *hardware*/perangkat keras yang sangat murah, dan tidak membutuhkan perangkat keras dan lunak dengan spesifikasi khusus. Arsitektur tersebut juga handal karena masing-masing komponen mempunyai memori dan disk tersendiri. MySQL Cluster menggabungkan MySQL Server biasa dengan sebuah mesin penyimpanan *in-memory* ter-*cluster* yang dinamakan NDB. NDB berarti bagian dari suatu rangkaian yang dikhususkan sebagai mesin penyimpanan, sedangkan MySQL Cluster diartikan sebagai kombinasi atau gabungan dari MySQL dan mesin penyimpanan yang baru tersebut.

2.11.1 Arsitektur MySQL Cluster

MySQL Cluster merupakan sebuah database yang menggunakan arsitektur *shared-nothing* dan antarmuka SQL yang telah umum digunakan. Sistem *database* ini terdiri dari beberapa node yang dapat didistribusikan ke beberapa perangkat keras dan ke beberapa wilayah/zona yang berbeda sekaligus untuk tetap menjaga ketersediaan data meskipun jaringan ataupun salah satu *node* sedang mengalami kegagalan (*failure*). Ada tiga *node* yang menyusun *MySQL Cluster*, yakni:

1. *Data Nodes*, digunakan untuk menyimpan semua data yang menjadi milik *MySQL Cluster*. Semua data direplikasi di node-node ini.

2. *Management Server Nodes*, digunakan untuk mengendalikan konfigurasi sistem ketika *startup*. Selain itu, *node* ini juga dapat digunakan sebagai pengidentifikasi setiap perubahan setting yang terjadi pada *cluster*.
3. *MySQL Server Nodes*, berfungsi sebagai pintu akses untuk masuk ke dalam *node-node data* yang ter-*cluster*.

2.12 Role Based Access Control (RBAC)

RBAC adalah sistem yang diterapkan pada aplikasi yang berhubungan dengan pengontrolan akses sumber daya. RBAC memberikan hak akses untuk peran (*roles*). Perancang kebijakan atau *administrator* sangat berperan dalam memberikan hak kepada para pelaku, sehingga subjek akan mendapatkan akses ke objek melalui *role* yang telah diberikan oleh *administrator*. Fitur yang disediakan oleh RBAC menjadi daya tarik bagi para perusahaan yang menerapkan teknologi informasi, pertama RBAC yang mengorganisir subjek dan *role* secara alamiah sesuai dengan struktur yang diterapkan pada perusahaan tersebut. Hubungan antara hak akses dengan para pelaku yang diterapkan pada perusahaan tersebut, pertama RBAC memberikan tugas keamanan pada kontrol akses sebagai prioritas tertinggi untuk mengontrol akses ke sumber daya. Hal tersebut mengakibatkan RBAC akan menerapkan keamanan yang sangat ketat dalam melakukan kontrol akses ke sumber daya. Kedua RBAC dalam menerapkan hak akses kepada pengguna membutuhkan waktu yang singkat, dengan cara menghubungkan subjek dengan *role*, sehingga memerlukan penunggasan hak akses untuk role pada setiap subjek.

Kontrol akses dalam mengambil keputusan ditentukan oleh *role*, sehingga pengguna sebagai bagian dari sebuah organisasi akan mendapatkan hak akses sesuai dengan *role* yang didapatkannya. Kebijakan yang dilakukan oleh RBAC akan membuat control akses yang didapatkan oleh pengguna berdasarkan keputusan yang diperoleh dalam sebuah organisasi. Pengguna tidak bisa mengambil hak akses pengguna lain, hal inilah dasar perbedaan antara RBAC dan DAC.

Dalam menangani kebijakan keamanan, RBAC adalah fondasi utama yang harus didefinisikan dengan baik sebelum *Security*

Constraint Specifiers (SCS) atau disebut juga sebagai petugas yang menentukan kendala dalam menjaga keamanan. SCS ini akan menentukan keterbatasan yang terjadi seperti kendala dalam penerapan sistem. RBAC memiliki banyak variasi yang akan dilakukan dengan menggambarkan bagaimana menentukan keterbatasan yang terjadi tanpa kehilangan ciri khasnya. Struktur RBAC meliputi *roles, permission, user, dan session*.

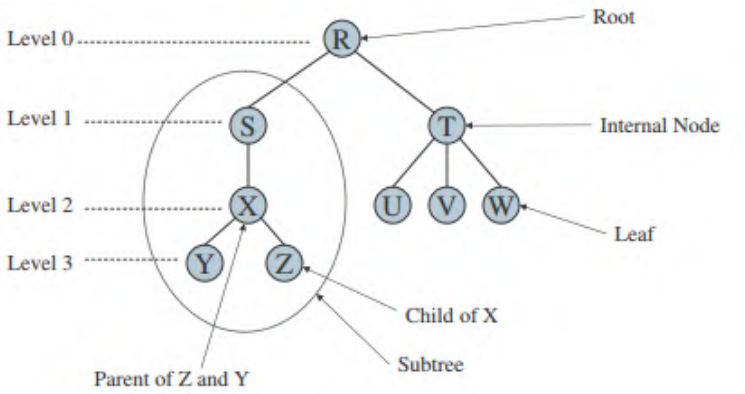
Dalam RBAC, *role* didefinisikan sebagai suatu gagasan yang merupakan dasar dari kebijakan kontrol akses. Pendefinisian *role* masih menjadi perdebatan sehingga diperlukan menjelasan konsep *role* itu tersendiri, sehingga untuk melakukan kolaborasi dengan kontrol akses masih disesuaikan dengan kebutuhan. Dalam ilmu perilaku, *role* didefinisikan sebagai pola yang ditentukan sesuai perilaku seseorang dalam situasi tertentu berdasarkan posisi orang tersebut. Dengan kata lain pendefinian *role* adalah tugas yang didapatkan oleh seseorang sesuai dengan tanggung jawabnya.

2.13 Algoritma Tree

Kumpulan *node* yang saling terhubung satu sama lain dalam suatu kesatuan yang membentuk layakya struktur sebuah pohon. Struktur pohon adalah suatu cara merepresentasikan suatu struktur hirarki (*one-to-many*) secara grafis yang mirip sebuah pohon, walaupun pohon tersebut hanya tampak sebagai kumpulan node-node dari atas ke bawah. Suatu struktur data yang tidak linier yang menggambarkan hubungan yang hirarkis (*one-to-many*) dan tidak linier antara elemen-elemennya.

Contoh penggunaan Struktur *Tree*:

- Silsilah keluarga
- Hasil pertandingan bentuk turnamen
- Struktur organisasi perusahaan



Gambar 2.1. Contoh Struktur Tree

2.13.1 Binary Tree

Binary Tree adalah *Tree* dengan syarat bahwa tiap *node* hanya boleh memiliki maksimal dua subpohon dan kedua subpohon harus terpisah.

Kelebihan struktur Binary Tree :

- Mudah dalam penyusunan algoritma *sorting*
- *Searching* data relatif cepat
- Fleksibel dalam penambahan dan penghapusan data

Pembentukan Binary Tree :

- Dapat dilakukan dengan dua cara : rekursif dan non rekursif
- Perlu memperhatikan kapan suatu node akan dipasang sebagai *node* kiri dan kapan sebagai *node* kanan
- Misal ditentukan, node yang berisi info yang nilainya ”lebih besar” dari *parent* akan ditempatkan disebelah kanan dan yang “lebih kecil disebelah kiri”.

Algoritma *Binary Tree* :

1. Buat *node* baru

2. Cek apakah *root* = *null*,
   Jika ya, maka *root* = *new* melompat ke langkah 9;
   Jika tidak, maka lakukan langkah berikut:
3. Mencari posisi yang tepat untuk *new*, tentukan p = *root*, q = *root*
4. Kerjakan langkah 5 & 6 selama q <> *null* dan *new* → info <> p → info
5. Tentukan p = q
6. Cek apakah new → info < p → info
   Jika ya, (teruskan ke cabang kiri), tentukan q = p → kiri;
   Jika tidak, (teruskan ke kanan), tentukan q = p → kanan;
7. Cek apakah *new* → info = p → info
   Jika ya, (tidak perlu disisipkan), tampilkan pesan duplikasi, lompat ke langkah 9.
   Jika tidak, (sisipkan), kerjakan langkah 8
8. Cek apakah new → info < p → info
   Jika ya, (sebagai cabang kiri), p → kiri = *new*
   Jika tidak, (sebagai cabang kanan), p → kanan = *new*
9. Selesai

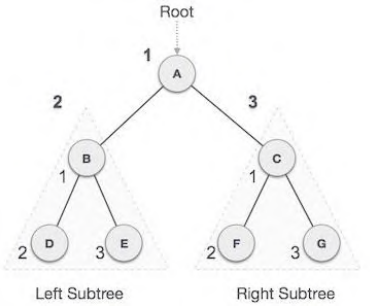
2.13.2 Tree Traversal Algorithm

Binary tree traversal algorithm adalah algoritma dari proses mengunjungi node tepat satu kali dan tiap node hanya boleh memiliki maksimal 2 *subtree* yang disebut sebagai sub pohon kiri (*left subtree*) dan sub pohon kanan (*right subtree*).

Dengan melakukan kunjungan secara lengkap, maka akan didapatkan urutan informasi secara linier yang tersimpan dalam sebuah *binary tree*.

2.13.2.1 Preorder Traversal Algorithm

Pada metode *traversal* ini, *node* paling pertama akan dikunjungi terlebih dahulu, setelah itu dilanjutkan dengan cabang bagian kiri, lalu cabang bagian kanan.

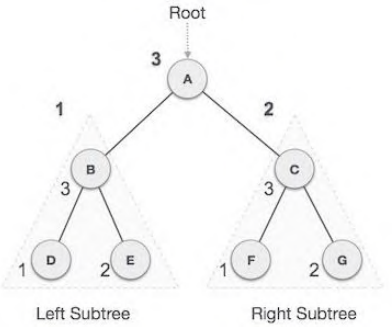


Gambar 2.2. Contoh *Preorder Traversal Algorithm*

Pada Gambar 2.2, akan dimulai dari *node* A, apabila dijalankan menggunakan algoritma *preorder traversal*, maka setelah A dikunjungi, algoritma tersebut akan mengunjungi B. Apabila pada *node* B juga dijalankan algoritma *preorder traversal*, maka yang selanjutnya akan dikunjungi adalah D, dan seterusnya. Sehingga keluaran dari algoritma *preorder traversal* adalah **A → B → D → E → C → F → G**

2.13.2.2 Postorder Traversal Algorithm

Pada metode traversal ini, root paling utama akan dikunjungi terakhir. Yang akan pertama dikunjungi adalah *node* paling kiri-bawah.



Gambar 2.3. Contoh *Postorder Traversal Algorithm*

Pada Gambar 2.3, apabila dijalankan dengan algoritma *postorder traversal*, maka akan dimulai dari *node* paling kiri-bawah yaitu *node* D. Setelah itu, *node* D akan mengunjungi *sibling* dari dirinya yaitu E. Jika tidak ada *sibling* lainnya, maka akan dilanjutkan ke *parent*-nya. Lalu *node* B memiliki *sibling* C, namun *node* C memiliki *child* (F dan G), oleh karena itu dilanjutkan oleh *child*-nya dahulu baru kemudian ke *parent*-nya kembali yaitu *node* C. Sehingga keluaran dari *postorder traversal* adalah **D → E → B → F → G → C → A**

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini akan dibahas tahap analisis dan perancangan sistem yang akan dibangun. Analisis membahas semua persiapan yang akan menjadi pokok pikiran pembuatan aplikasi ini. Mulai dari masalah yang melatarbelakangi, hingga analisis gambaran awal sistem yang akan dibuat. Perancangan sistem membahas hal-hal yang berkaitan dengan pondasi atau dasar pembuatan aplikasi, yang meliputi perancangan basis data, tampilan antar muka halaman aplikasi, hingga perancangan alur proses yang akan diimplementasikan di dalam aplikasi.

3.1 Analisis

Tahap analisis dibagi menjadi beberapa bagian antara lain: domain permasalahan, deskripsi umum sistem, dan kasus penggunaan sistem. Berikut pembahasan bagian-bagian tahap analisis.

3.1.1 Analisis Proses Bisnis

Aplikasi ERP bukan aplikasi yang baru muncul atau jarang digunakan. Aplikasi ERP merupakan aplikasi yang sudah lama ada. Namun masing-masing ERP memiliki proses bisnis yang berbeda-beda. Kebanyakan aplikasi ERP yang kompleks dan sesuai dengan banyak perusahaan memiliki harga jual yang tinggi. Permasalahan berada pada bagaimana *business plan* yang telah ada dapat dijalankan secara efisien dengan aplikasi ERP ini. Setiap ERP memiliki proses bisnis yang berbeda-beda. Aplikasi ERP pada tugas akhir ini memiliki proses bisnis sendiri dan dibandingan dengan aplikasi ERP yang cukup banyak digunakan banyak perusahaan. Seperti Odoo, Adempiere, dan InoERP. Pada bab ini akan dijelaskan tentang analisa proses bisnis yang telah ada. Proses bisnis yang dimiliki oleh Odoo, Adempiere, dan InoERP ditunjukkan pada Lampiran A.1, Lampiran A.2, dan Lampiran A.3. Setelah ditelaah lebih dalam, terdapat beberapa kekurangan dan kelebihan proses bisnis yang dimiliki oleh masing-masing ERP

khususnya pada. Kekurangan dan kelebihan tersebut akan dijelaskan pada tabel 3.1.

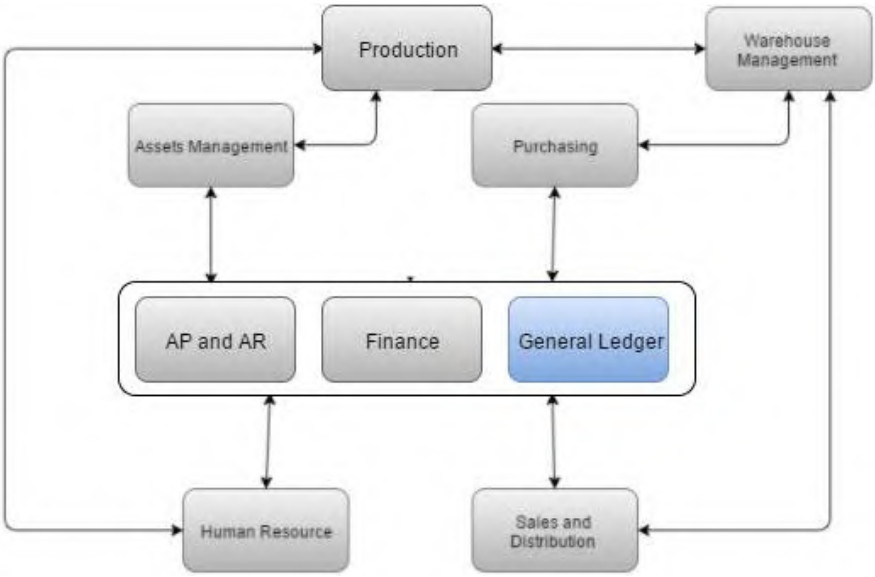
Tabel 3.1. Kekurangan dan kelebihan pada Odoo, Adempiere, dan InoERP pada modul General Ledger

Nama Modul	**Odoo**	**Adempiere**	**InoERP**
General Ledger	**Pro** : Pembuatan akun yang terbilang simple sehingga mudah digunakan. **Kontra :** Jurnal masih dapat dibuat secara manual biarpun dalam kondisi belum balance	**Kontra :** Pembuatan Journal yang rumit dikarenakan harus save header dahulu baru linenya	**Pro :** Masing-masing modul sudah menyimpan akun yang akan digunakan sehingga mempermudah proses penjurnalan **Kontra :** Pembuatan akun yang terbilang rumit, sehingga diaplikasikan algoritma tree dalam pembuatan akun.

Tabel 3.1 menunjukkan bahwa masih banyak kekurangan terutama dalam sisi kemudahan dan efisiensi. Seperti pada Odoo, pembuatan akun bisa terbilang cukup mudah dan tidak memakan waktu, namun untuk pembuatan jurnal merupakan hal yang tidak bisa ditoleransi, karena dapat berakibat fatal di akhir. Untuk Adempiere, terdapat kekurangan khususnya pada sisi kemudahan, karena sulitnya membuat jurnal pada sistem ERP tersebut. Pada sisi inoERP, cara kerja keseluruhan ERP dan kebergantungan akun sangat dibutuhkan, namun

hal ini bagus karena dapat meminimalisir kesalahan, biarpun begitu, cara pembuatan akun pada inoERP terbilang rumit, sehingga tidak efisien.

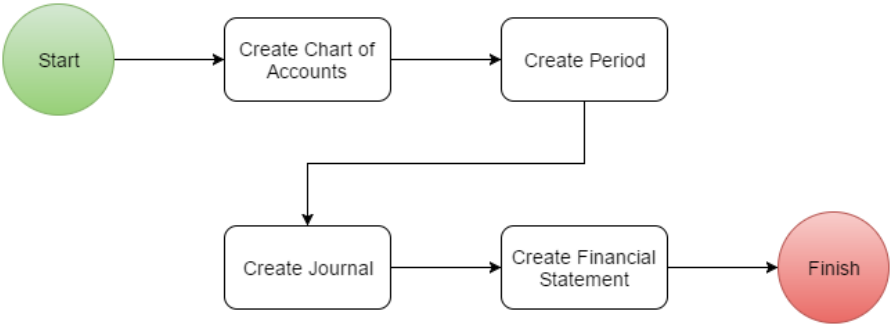
Berdasarkan kekurangan yang dimiliki oleh setiap ERP maka perlu dikembangkannya sebuah aplikasi ERP yang sesuai dengan kebutuhan pada *business plan.* Terdapat dua jenis proses bisnis yang digunakan, yakni *Make-To-Order* (MTO) dan *Make-To-Stock* (MTS). MTO adalah sebuah proses bisnis dimana produksi didasarkan pada permintaan yang dilakukan oleh *sales order*. MTS adalah sebuah proses bisnis dimana produksi didasarkan oleh peramalan penjualan. Proses bisnis yang dirancang sesuai dengan kebutuhan *business plan* dimana proses bisnis ini telah mengakomodasi kekurangan Odoo, Adempiere, dan InoERP ditunjukkan pada Gambar 3.1, Gambar 3.2, Gambar 3.3, Gambar 3.4, dan Gambar 3.5.



Gambar 3.1. Proses Bisnis ERP Level 0

Pada Gambar 3.1, modul-modul keuangan (AP/AR, *Finance*, dan *General Ledger*) merupakan sentral dari sistem ERP. Sedangkan, modul *Sales and Distribution* berinteraksi dengan modul *Inventory and Warehouse Management* dan modul *Assets Management*.

Proses bisnis ERP level 0 dapat dipecah kembali menjadi proses-proses bisnis yang lebih rinci berdasarkan subbab 2.2. Proses bisnis secara keseluruhan ditunjukkan pada Lampiran A.4 dan Lampiran A.5. Hasil pemecahan proses bisnis ditunjukkan pada Gambar 3.2.



Gambar 3.2. Proses Bisnis *Level* 1 *Make-to-Stock* dan *Make-to-Order*

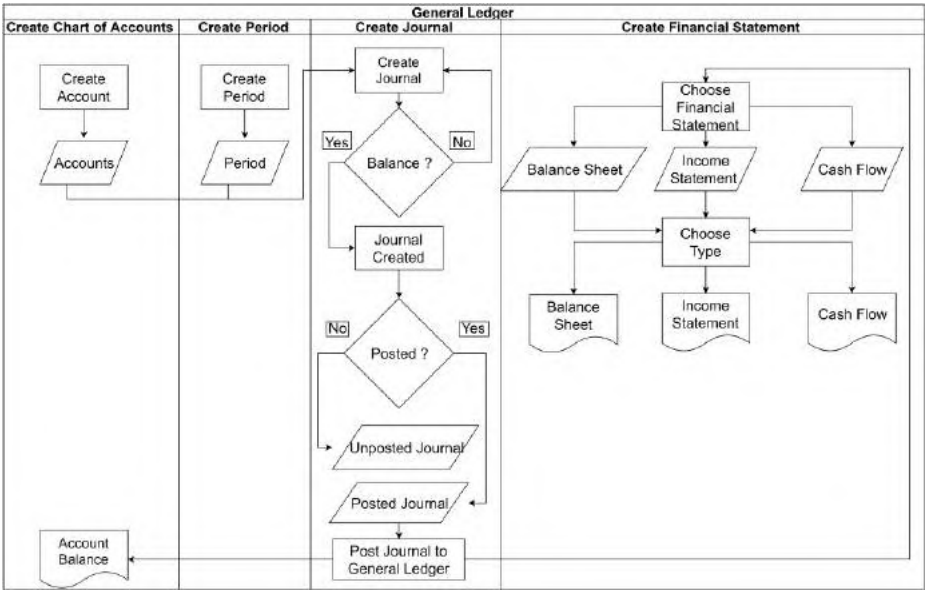
Pada Gambar 3.2 ditunjukkan proses bisnis *level* 1 MTO dan MTS. Penjelasan proses bisnis *level* 1 terdapat pada Tabel 3.2.

Tabel 3.2. Deskripsi Proses Bisnis *Level* 1 Modul *General Ledger*

No	Proses Bisnis	Keterangan
1	*Create Chart of Accounts*	Merupakan proses bisnis pembuatan *Chart of Accounts*
2	*Create Period*	Merupakan proses bisnis pembuatan *Period*
3	*Create Journal*	Merupakan proses bisnis pembuatan *Journal*

4	*Create Financial Statement*	Merupakan proses bisnis pembuatan *Financial Statement* (Laporan Keuangan)

Dari masing-masing proses bisnis *level* 1 dapat dipecah kembali menjadi proses bisnis yang lebih rinci lagi. Proses bisnis ditunjukkan pada Gambar 3.3.

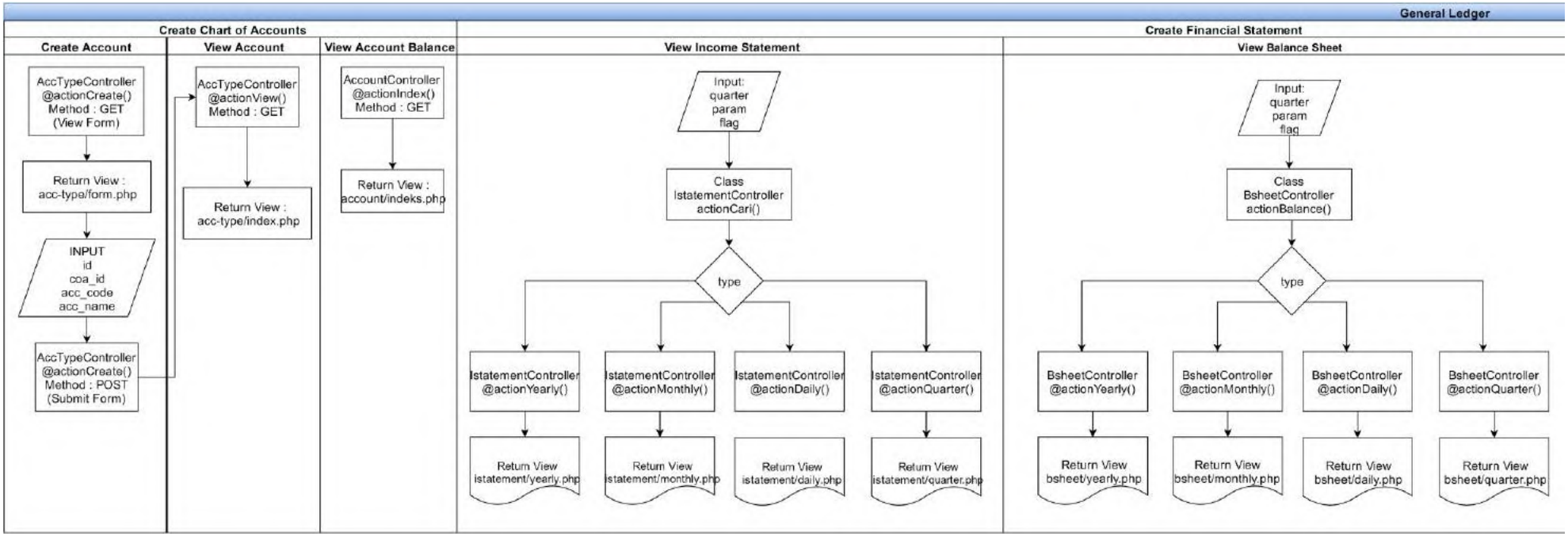


Gambar 3.3. Proses Bisnis *Level* 2 *Make-to-Stock* dan *Make-to-Order*

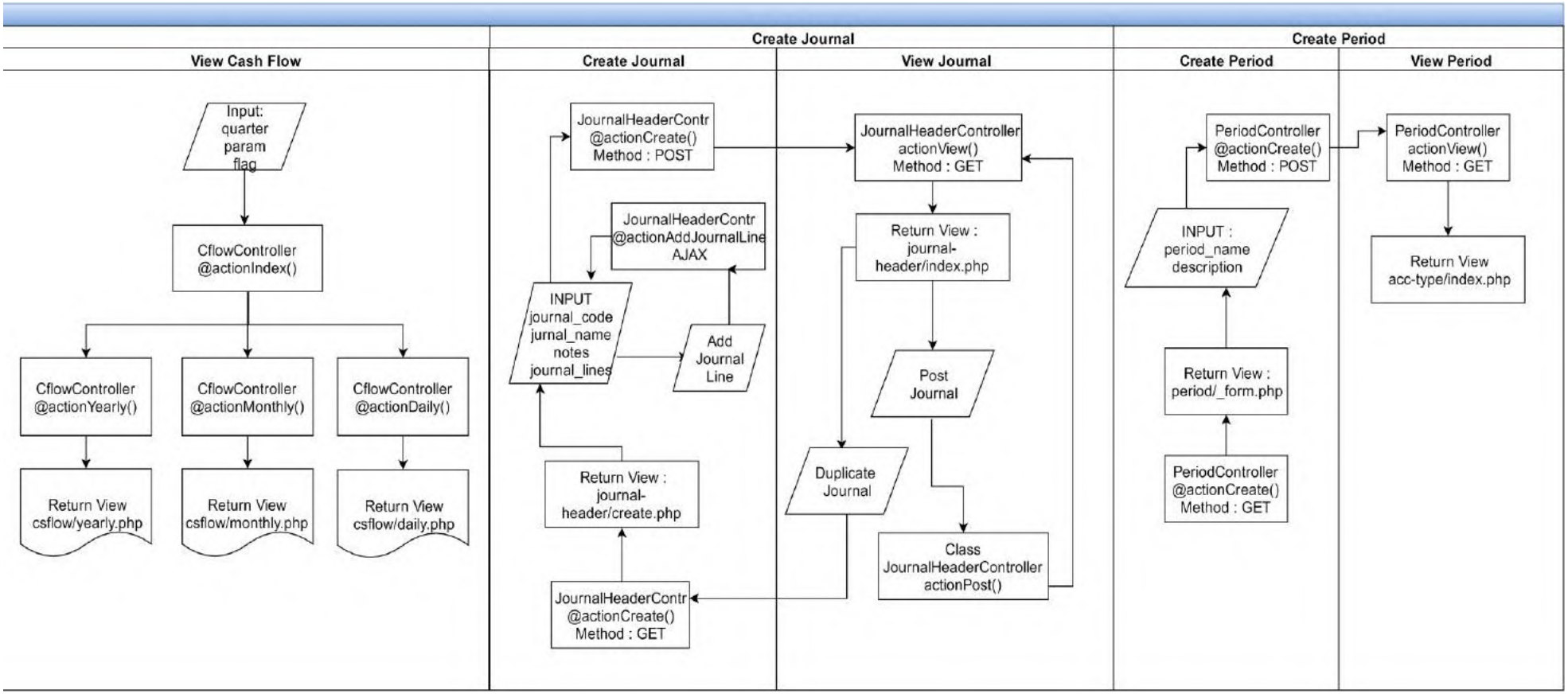
Pada Gambar 3.3 ditunjukkan proses bisnis *level* 2 MTO dan MTS. Penjelasan proses bisnis *level* 2 terdapat pada Tabel 3.3.

Tabel 3.3. Deskripsi Proses Bisnis *Level* 2 Modul *General Ledger*

No	Proses Bisnis	Keterangan
1	*Create Account*	Merupakan proses bisnis pembuatan akun
2	*Create Period*	Merupakan proses bisnis pembuatan periode akuntansi
3	*Create Journal*	Merupakan proses bisnis pembuatan jurnal keuangan
4	*Journal Created*	Merupakan proses bisnis dimana jurnal telah terbuat
5	*Unposted Journal*	Dokumen berupa jurnal yang tidak di-*post* ke dalam laporan keuangan
6	*Posted Journal*	Dokumen berupa jurnal yang akan di-*post* ke dalam laporan keuangan
7	*Post Journal to General Ledger*	Merupakan proses bisnis untuk memasukkan jurnal ke dalam laporan keuangan
8	*Choose Financial Statement*	Merupakan proses bisnis pemilihan laporan keuangan yang akan ditampilkan
9	*Choose Type*	Merupakan proses bisnis pemilihan jenis laporan keuangan (tahunan, bulanan, harian)



Gambar 3.4. Proses Bisnis *Make-to-Order* dan *Make-to-Stock* Modul *General Ledger* (bag. 1)



Gambar 3.5. Proses Bisnis *Make-to-Order* dan *Make-to-Stock* Modul *General Ledger* (bag. 2)

Dari masing-masing proses bisnis *level* 2 dapat dipecah kembali menjadi proses bisnis yang lebih rinci lagi. Proses bisnis ditunjukkan pada Gambar 3.4.

Gambar 3.4 dan 3.5 merupakan proses bisnis *level* 3 MTO dan MTS. Pada proses bisnis *level* 3 menjelaskan detail dari kelas, fungsi, dan halaman yang ditampilkan untuk melakukan proses bisnis yang terdapat pada *level* 2.

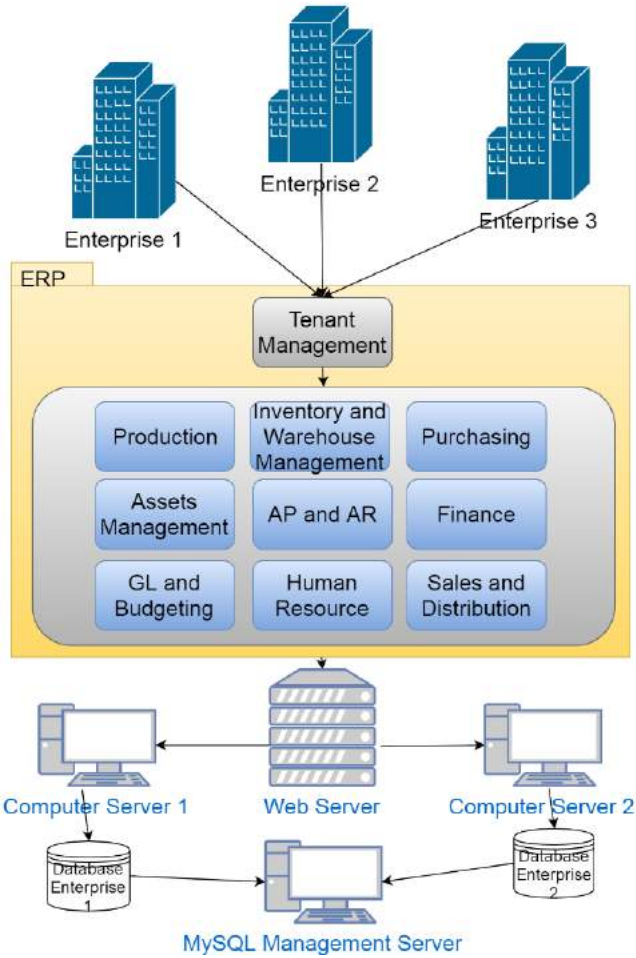
3.1.2 Analisis Data

Konsep autentikasi dan otorisasi data yang digunakan adalah *multitenancy separated schema*. Pada konsep ini, data milik setiap *tenant* akan disimpan di basis data yang terpisah. Sehingga tidak ada 2 atau lebih *tenant* menggunakan *table* dan basis data yang sama. Penambahan *tenant* baru akan menyebabkan penambahan sebuah basis data baru. Penggunaan konsep *separated database* dimaksudkan agar data milik setiap *tenant* dapat terjamin kerahasiaannya. Keuntungan lainnya adalah kerumitan proses *maintenance* basis data dapat dikurangi.

Sedangkan konsep basis data terdistribusi yang diterapkan adalah replikasi. Replikasi memungkinkan penyalinan setiap tabel basis data ke *node-node* penyusun basis data terdistribusi. Kegagalan sebuah *node* tidak akan menyebabkan basis data berhenti bekerja. Sebaliknya, hal tersebut akan memicu mekanisme sinkronisasi jika *node* yang mati kembali hidup.

3.2 Deskripsi Umum Sistem

Pada Tugas Akhir ini dibangun aplikasi *Sales and Distribution* yang terintegrasi ke dalam sebuah sistem *Enterprise Resource Planning* (ERP). Tujuan dari aplikasi *Sales and Distribution* adalah untuk mengelola dan melakukan monitoring terhadap penjualan dan pelanggan. Sistem yang dibangun berorientasi *multitenancy* dengan basis data terdistribusi. Desain basis data terdistribusi dan *multitenancy* yang digunakan dalam sistem ini ditunjukkan pada Gambar 3.6.



Gambar 3.6. Rancang Basis Data Terdistribusi

Pada Gambar 3.6, dijelaskan bahwa perusahaan 1, perusahaan 2, dan perusahaan 3 dapat menggunakan ERP secara bersamaan. Setiap perusahaan mempunyai 2 basis data yang telah direplikasi yaitu *Computer Server 1* dengan IP *Enterprise* 1 dan *Computer Server 2* dengan IP *Enterprise* 2. Komputer 3 berfungsi sebagai *server cluster*

untuk melakukan replikasi, sehingga apabila basis data 1 atau *Computer Server 1* dalam keadaan mati, sistem ERP masih tetap berjalan.

3.3 Aktor

Pada sistem yang akan dibangun, aktor yang akan menjadi pengguna sistem adalah staff dan manager *accounting*. Staff asset mempunyai hak ases dalam pengelolaan aset, yang terdiri atas membuat (*create*), memperbarui (*update*), melihat (*view*) dan menghapus (*delete*). Sedangkan manager mempunyai hak akses melihat (*view*) data yang telah dikelola oleh staff.

3.4 Spesifikasi Kebutuhan Perangkat Lunak

Spesifikasi kebutuhan dalam sistem ini mencakup kebutuhan fungsional. Kebutuhan fungsional berisikan proses-proses yang dibutuhkan dalam sistem dan harus dijalankan. Kebutuhan fungsional sistem dideskripsikan dalam Tabel 3.4.

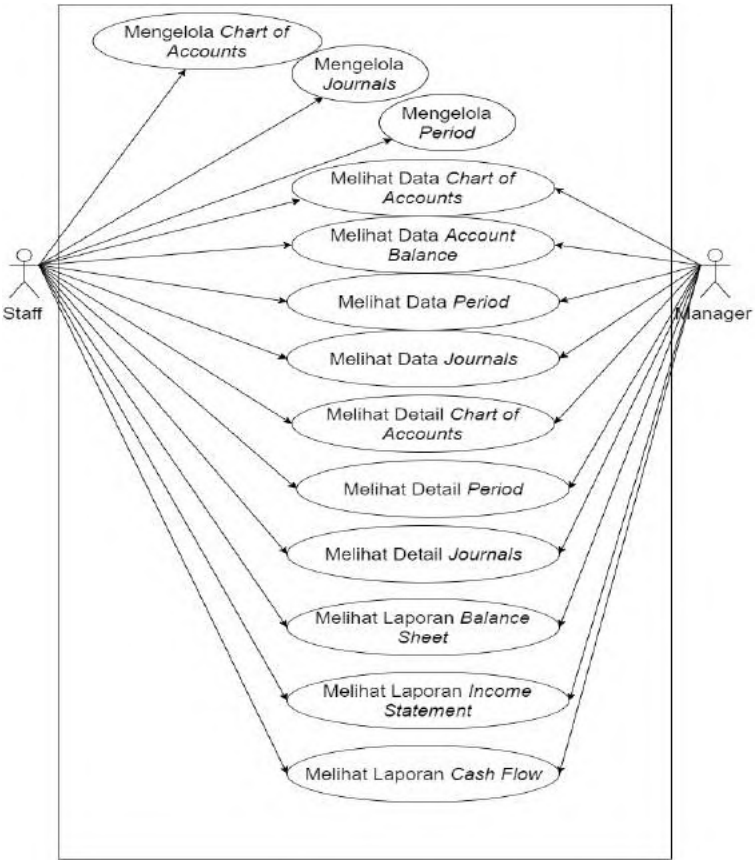
Tabel 3.4. Daftar Kebutuhan Fungsional Sistem

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
F-001	Mengelola *Chart of Accounts*	Pengguna dapat mengelola *chart of accounts* yang meliputi proses: *create, update, delete*
F-002	Mengelola *Period*	Pengguna dapat mengelola *period* yang meliputi proses: *create, update, delete*
F-003	Mengelola *Journals*	Pengguna dapat mengelola *journals* yang meliputi proses: *create, update, delete*
F-004	Melihat Data *Chart of Accounts*	Pengguna dapat melihat data *Chart of Accounts*

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
F-005	Melihat Data *Account Balance*	Pengguna dapat melihat data *Account Balance*
F-006	Melihat Data *Period*	Pengguna dapat melihat data *Period*
F-007	Melihat Data *Journals*	Pengguna dapat melihat data *Journals* baik *posted* maupun *unposted*
F-008	Melihat Detail *Chart of Accounts*	Pengguna dapat melihat data detail per *chart of account*
F-009	Melihat Detail *Period*	Pengguna dapat melihat data detail per *period*
F-010	Melihat Detail *Journals*	Pengguna dapat melihat data detail per *journal*
F-011	Melihat Laporan *Balance Sheet*	Pengguna dapat melihat laporan *Balance Sheet* secara tahunan, bulanan, maupun harian
F-012	Melihat Laporan *Income Statement*	Pengguna dapat melihat laporan *Income Statement* secara tahunan, bulanan, maupun harian
F-013	Melihat Laporan *Cash Flow*	Pengguna dapat melihat laporan *Cash Flow* secara tahunan, bulanan, maupun harian

3.5 Kasus Penggunaan

Kasus penggunaan yang dibutuhkan pada sistem sesuai dengan analisa yang telah dilakukan. Pada sistem ERP 2015 modul *accounting* terdapat 13 kasus penggunaan dan terdapat 2 aktor. Diagram kasus penggunaan dapat dilihat pada Gambar 3.7 dan kode kasus penggunaan ada pada tabel 3.5



Gambar 3.7. Diagram Kasus Penggunaan

Tabel 3.5. Daftar Kasus Penggunaan

No.	Kode	Nama Kasus Penggunaan	Aktor
1	UC-001	Mengelola *Chart of Accounts*	Staff
2	UC-002	Mengelola *Journals*	Staff
3	UC-003	Mengelola *Period*	Staff
4	UC-004	Melihat Data *Chart of Accounts*	Staff, Manager
5	UC-005	Melihat Data *Account Balance*	Staff, Manager
6	UC-006	Melihat Data *Period*	Staff, Manager
7	UC-007	Melihat Data *Journals*	Staff, Manager
8	UC-008	Melihat Detail *Chart of Accounts*	Staff, Manager
9	UC-009	Melihat Detail *Period*	Staff, Manager
10	UC-010	Melihat Detail *Journals*	Staff, Manager
11	UC-011	Melihat Laporan *Balance Sheet*	Staff, Manager
12	UC-012	Melihat Laporan *Income Statement*	Staff, Manager
13	UC-013	Melihat Laporan *Cash Flow*	Staff, Manager

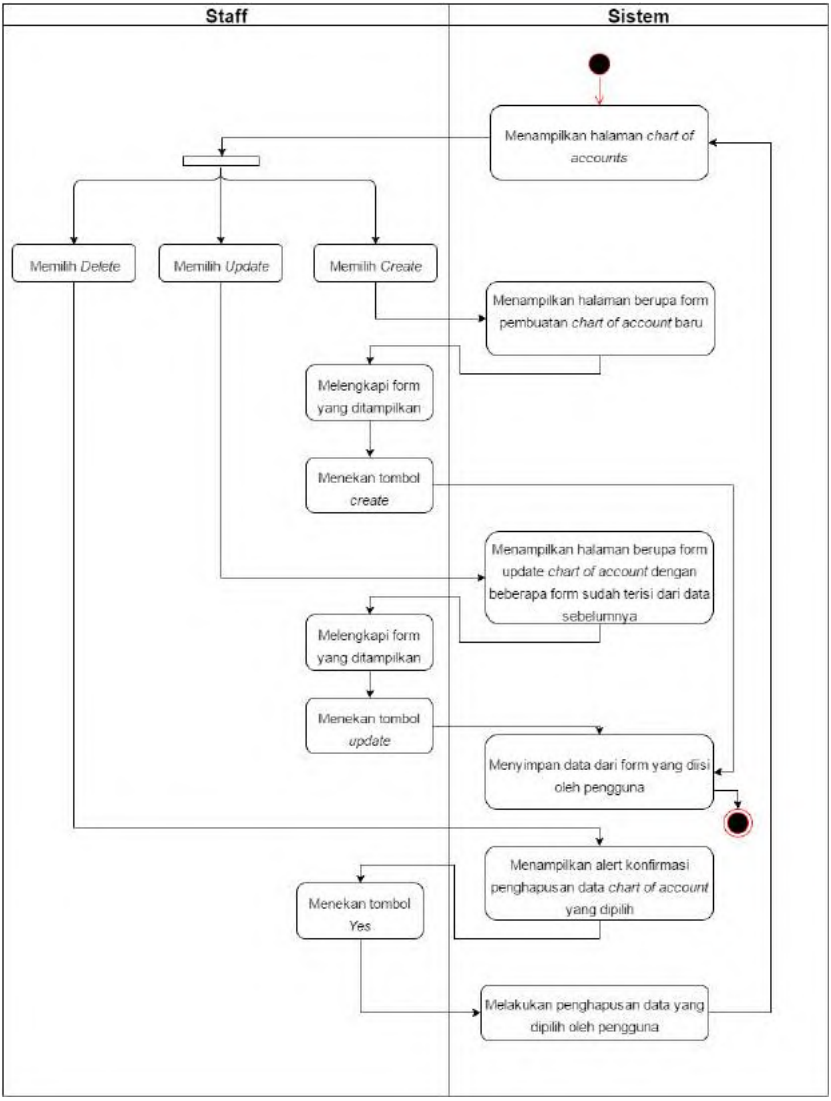
3.5.1 Kasus Penggunaan Mengelola *Chart of Accounts*

Pada kasus penggunaan ini, sistem menampilkan halaman untuk menambahkan, mengubah, dan menghapus *chart of accounts*. Pengguna dapat menambahkan, mengubah, dan menghapus pada tampilan yang tersedia. Tabel kasus penggunaan mengelola *chart of accounts* dapat dilihat pada Tabel 3.6, sedangkan gambar diagram aktivitas dapat dilihat pada Gambar 3.8.

Tabel 3.6. Kasus Penggunaan Mengelola *Chart of Accounts*

Nama	**Mengelola *Chart of Accounts***
Nomor	UC-001
Deskripsi	Kasus penggunaan ini digunakan untuk mengelola data *Chart of Accounts*
Tipe	Fungsional
Aktor	*Staff*
Kondisi Awal	Pengguna sudah login dan masuk ke dalam sistem

Kondisi Akhir	Adanya perubahan pada data *Chart of Accounts*
Alur Normal	1. Sistem menampilkan halaman *Chart of Accounts* 2. Pengguna memilih *Create* A1. Pengguna memilih *Update* A2. Pengguna memilih *Delete* 3. Sistem menampilkan halaman berupa form pembuatan *chart of account* baru 4. Pengguna melengkapi form yang ditampilkan 5. Pengguna menekan tombol *Create* A3. Pengguna menekan tombol *Cancel* 6. Sistem menyimpan data dari form yang diisi oleh pengguna 7. Sistem menampilkan data *chart of account* yang sebelumnya diisi oleh pengguna
Alur Alternatif	A1. Pengguna memilih *Update* 1. Sistem menampilkan halaman berupa form update *chart of account* dengan beberapa form sudah terisi dari data sebelumnya 2. Pengguna melakukan perubahan terhadap form yang ditampilkan 3. Pengguna menekan tombol *Update* A3. Pengguna menekan tombol *Cancel* 4. Kembali ke kejadian 6 alur normal A2. Pengguna memilih *Delete* 1. Sistem menampilkan alert konfirmasi penghapusan data *chart of account* yang dipilih 2. Pengguna menekan tombol *Yes* A3. Pengguna menekan tombol *Cancel* *3.* Sistem melakukan penghapusan data yang dipilih oleh pengguna A3. Pengguna menekan tombol *Cancel* 1. Sistem kembali ke kejadian 1 alur normal.



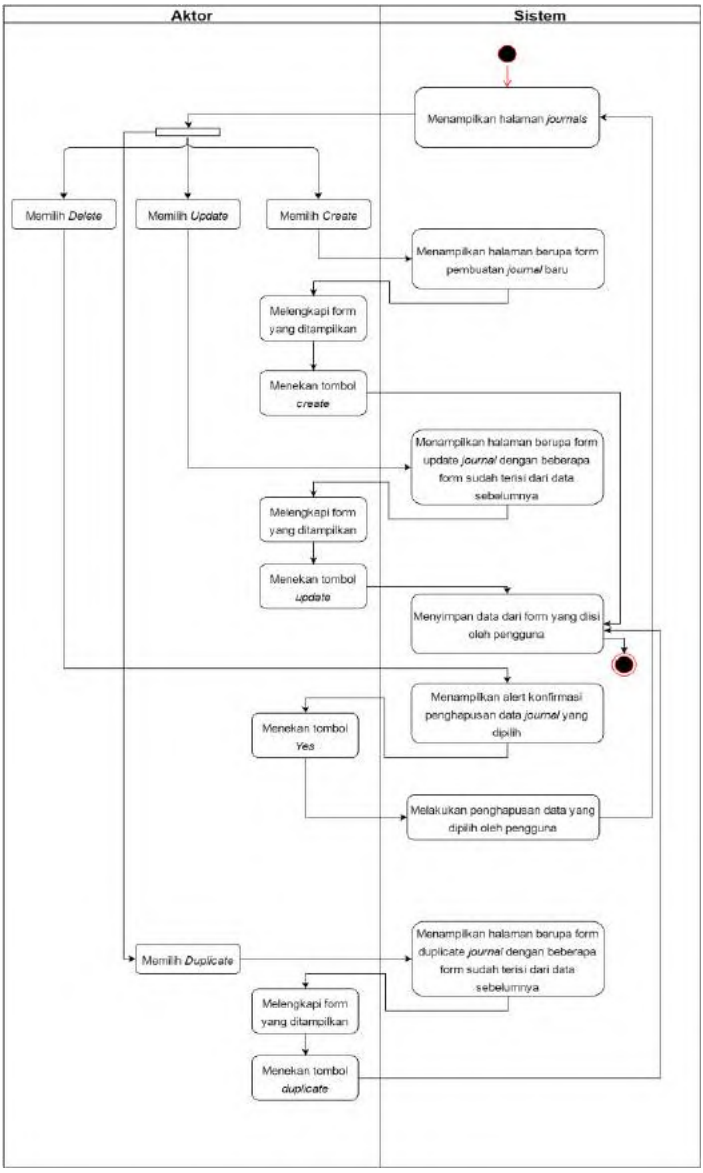
Gambar 3.8. Diagram Aktivitas Mengelola *Chart of Accounts*

3.5.2 Kasus Penggunaan Mengelola *Journals*

Pada kasus penggunaan ini, sistem menampilkan halaman untuk menambahkan, mengubah, menghapus, serta menduplikasi *journals*. Pengguna dapat menambahkan, mengubah, menghapus, dan menduplikasi pada tampilan yang tersedia. Tabel detil kasus penggunaan mengelola *journals* dapat dilihat pada Tabel 3.7, sedangkan penggambaran diagram aktivitas dapat dilihat pada Gambar 3.9.

Tabel 3.7. Kasus Penggunaan Mengelola *Journals*

Nama	**Mengelola *Journals***
Nomor	UC-002
Deskripsi	Kasus penggunaan ini digunakan untuk mengelola data *Journals*
Tipe	Fungsional
Aktor	*Staff*
Kondisi Awal	Pengguna sudah login dan masuk ke dalam sistem
Kondisi Akhir	Adanya perubahan pada data *Journals*
Alur Normal	1. Sistem menampilkan halaman *Journals* 2. Pengguna memilih *Create* A1. Pengguna memilih *Update* A2. Pengguna memilih *Delete* A4. Pengguna memilih *Duplicate* 3. Sistem menampilkan halaman berupa form pembuatan *journal* baru 4. Pengguna melengkapi form yang ditampilkan 5. Pengguna menekan tombol *Create* A3. Pengguna menekan tombol *Cancel* 6. Sistem menyimpan data dari form yang diisi oleh pengguna 7. Sistem menampilkan data *journal* yang sebelumnya diisi oleh pengguna

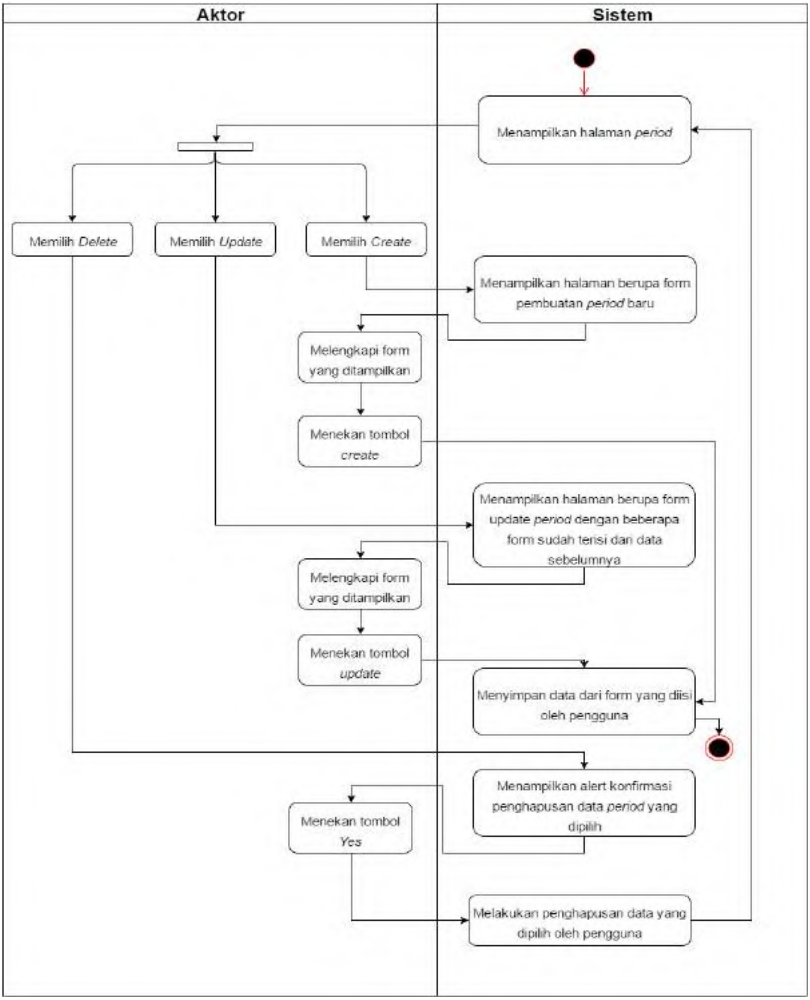


Gambar 3.9. Diagram Aktivitas Mengelola *Journal*

Alur Alternatif	A1. Pengguna memilih *Update* 1. Sistem menampilkan halaman berupa form update *journal* dengan beberapa form sudah terisi dari data sebelumnya 2. Pengguna melakukan perubahan terhadap form yang ditampilkan 3. Pengguna menekan tombol *Update* A3. Pengguna menekan tombol *Cancel* 4. Kembali ke kejadian 6 alur normal A2. Pengguna memilih *Delete* 1. Sistem menampilkan alert konfirmasi penghapusan data *journal* yang dipilih 2. Pengguna menekan tombol *Yes* A3. Pengguna menekan tombol *Cancel* *3.* Sistem melakukan penghapusan data yang dipilih oleh pengguna A3. Pengguna menekan tombol *Cancel* 1. Kembali ke kejadian 1 alur normal. A4. Pengguna memilih *Duplicate* 1. Sistem menampilkan halaman berupa form duplicate *journal* dengan beberapa form sudah terisi dari data sebelumnya sedangkan catatan dan detil dikosongkan 2. Pengguna melakukan perubahan terhadap form yang ditampilkan 3. Pengguna menekan tombol *Update* A3. Pengguna menekan tombol *Cancel* 4. Kembali ke kejadian 6 alur normal

3.5.3 Kasus Penggunaan Mengelola *Period*

Pada kasus penggunaan ini, sistem menampilkan halaman untuk menambahkan, mengubah, dan menghapus *period*. Pengguna dapat menambahkan, mengubah, dan menghapus pada tampilan yang tersedia. Tabel kasus penggunaan mengelola *period* dapat dilihat pada Tabel 3.8, sedangkan gambar diagram aktivitas dapat dilihat pada Gambar 3.10.



Gambar 3.10. Diagram Aktivitas Mengelola *Period*

Tabel 3.8. Kasus Penggunaan Mengelola *Period*

Nama	**Mengelola *Period***
Nomor	UC-003

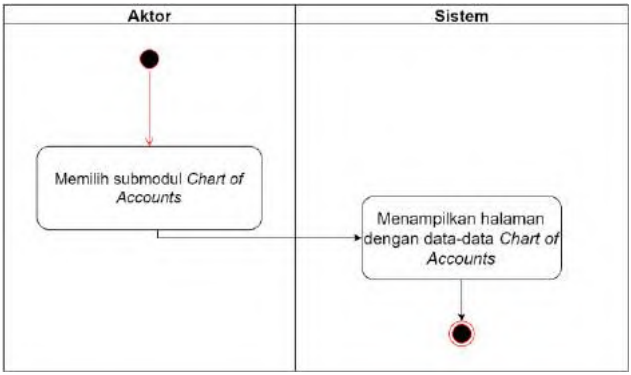
Deskripsi	Kasus penggunaan ini digunakan untuk mengelola data *Period*
Tipe	Fungsional
Aktor	*Staff*
Kondisi Awal	Pengguna sudah login dan masuk ke dalam sistem
Kondisi Akhir	Adanya perubahan pada data *Period*
Alur Normal	1. Sistem menampilkan halaman *Period* 2. Pengguna memilih *Create* A1. Pengguna memilih *Update* A2. Pengguna memilih *Delete* 3. Sistem menampilkan halaman berupa form pembuatan *period* baru 4. Pengguna melengkapi form yang ditampilkan 5. Pengguna menekan tombol *Create* A3. Pengguna menekan tombol *Cancel* 6. Sistem menyimpan data dari form yang diisi oleh pengguna 7. Sistem menampilkan data *period* yang sebelumnya diisi oleh pengguna
Alur Alternatif	A1. Pengguna memilih *Update* 1. Sistem menampilkan halaman berupa form update *period* dengan beberapa form sudah terisi dari data sebelumnya 2. Pengguna melakukan perubahan terhadap form yang ditampilkan 3. Pengguna menekan tombol *Update* A3. Pengguna menekan tombol *Cancel* 4. Kembali ke kejadian 6 alur normal A2. Pengguna memilih *Delete* 1. Sistem menampilkan alert konfirmasi penghapusan data *periodl* yang dipilih 2. Pengguna menekan tombol *Yes* A3. Pengguna menekan tombol *Cancel* *3.* Sistem melakukan penghapusan data yang dipilih oleh pengguna A3. Pengguna menekan tombol *Cancel* 1. Kembali ke kejadian 1 alur normal.

3.5.4 Kasus Penggunaan Melihat Data *Chart of Accounts*

Pada kasus penggunaan ini, sistem menampilkan halaman untuk melihat data *chart of accounts.* Pengguna dapat melihat data yang tersimpan pada tampilan yang tersedia. Tabel kasus penggunaan melihat data *chart of accounts* dapat dilihat pada Tabel 3.9, sedangkan gambar diagram aktivitas dapat dilihat pada Gambar 3.11.

Tabel 3.9. Kasus Penggunaan Melihat Data *Chart of Accounts*

Nama	**Melihat Data *Chart of Accounts***
Nomor	UC-004
Deskripsi	Kasus penggunaan ini digunakan untuk melihat data *Chart of Accounts*
Tipe	Fungsional
Aktor	*Staff, Manager*
Kondisi Awal	Pengguna sudah login dan masuk ke dalam sistem
Kondisi Akhir	Sistem menampilkan halaman berupa data dari *chart of account*
Alur Normal	1. Pengguna memilih sub-modul *Chart of Accounts* 2. Sistem menampilkan halaman berupa data dari *Chart of Accounts*
Alur Alternatif	-



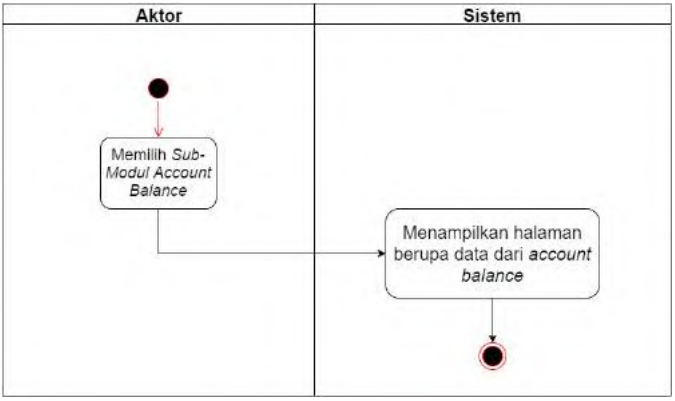
Gambar 3.11. Diagram Aktivitas Melihat Data *Chart of Accounts*

3.5.5 Kasus Penggunaan Melihat Data *Account Balance*

Pada kasus penggunaan ini, sistem menampilkan halaman untuk melihat data *Account Balance*. Pengguna dapat melihat data yang tersimpan pada tampilan yang tersedia. Tabel kasus penggunaan melihat data *Account Balance* dapat dilihat pada Tabel 3.10, sedangkan gambar diagram aktivitas dapat dilihat pada Gambar 3.12.

Tabel 3.10. Kasus Penggunaan Melihat Data *Account Balance*

Nama	**Melihat Data *Account Balance***
Nomor	UC-005
Deskripsi	Kasus penggunaan ini digunakan untuk melihat data *Account Balance*
Tipe	Fungsional
Aktor	*Staff, Manager*
Kondisi Awal	Pengguna sudah login dan masuk ke dalam sistem
Kondisi Akhir	Sistem menampilkan halaman berupa data dari *account balance*
Alur Normal	1. Pengguna memilih sub-modul *Account Balance* 2. Sistem menampilkan halaman berupa data dari *account balance*
Alur Alternatif	-



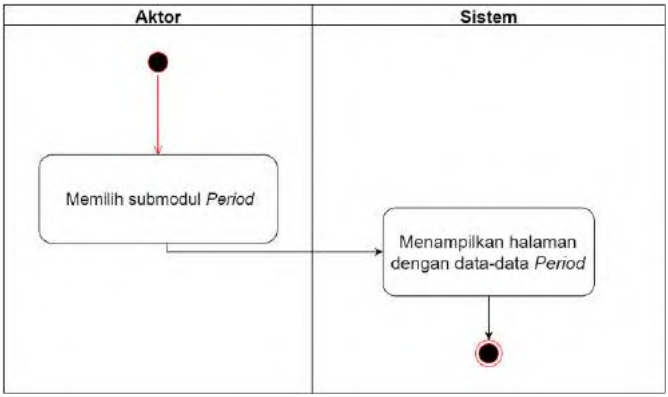
Gambar 3.12. Diagram Aktivitas Melihat Data *Account Balance*

3.5.6 Kasus Penggunaan Melihat Data *Period*

Pada kasus penggunaan ini, sistem menampilkan halaman untuk melihat data *Period.* Pengguna dapat melihat data yang tersimpan pada tampilan yang tersedia. Tabel kasus penggunaan melihat data *Period* dapat dilihat pada Tabel 3.11 sedangkan gambar diagram aktivitas dapat dilihat pada Gambar 3.13.

Tabel 3.11. Kasus Penggunaan Melihat Data *Period*

Nama	**Melihat Data *Period***
Nomor	UC-006
Deskripsi	Kasus penggunaan ini digunakan untuk melihat data *Period*
Tipe	Fungsional
Aktor	*Staff, Manager*
Kondisi Awal	Pengguna sudah login dan masuk ke dalam sistem
Kondisi Akhir	Sistem menampilkan halaman berupa data dari *period*
Alur Normal	1. Pengguna memilih sub-modul *Period* 2. Sistem menampilkan halaman berupa data dari *period*
Alur Alternatif	-



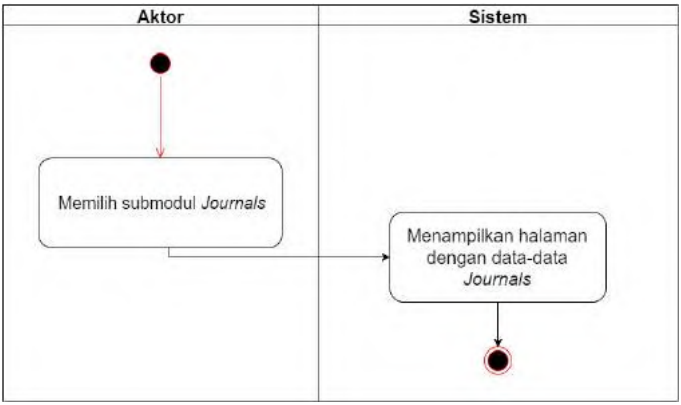
Gambar 3.13. Diagram Aktivitas Melihat Data *Period*

3.5.7 Kasus Penggunaan Melihat Data *Journal*

Pada kasus penggunaan ini, sistem menampilkan halaman untuk melihat data *Period.* Pengguna dapat melihat data yang tersimpan pada tampilan yang tersedia. Tabel kasus penggunaan melihat data *Period* dapat dilihat pada Tabel 3.12 sedangkan gambar diagram aktivitas dapat dilihat pada Gambar 3.14.

Tabel 3.12. Kasus Penggunaan Melihat Data *Journal*

Nama	**Melihat Data *Journals***
Nomor	UC-007
Deskripsi	Kasus penggunaan ini digunakan untuk melihat data *Journal*
Tipe	Fungsional
Aktor	*Staff, Manager*
Kondisi Awal	Pengguna sudah login dan masuk ke dalam sistem
Kondisi Akhir	Sistem menampilkan halaman berupa data dari *journal*
Alur Normal	1. Pengguna memilih sub-modul *Period* 2. Sistem menampilkan halaman berupa data dari *period*
Alur Alternatif	-



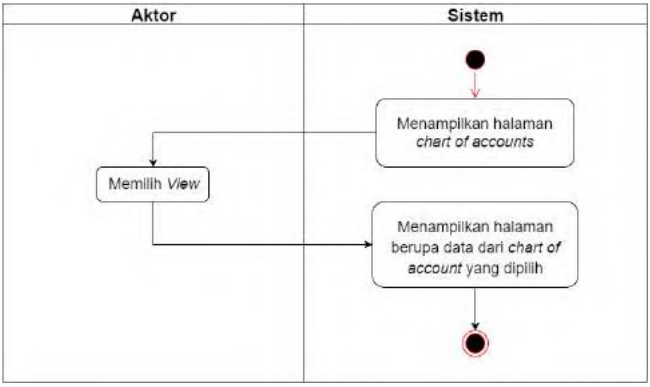
Gambar 3.14. Diagram Aktivitas Melihat Data *Journal*

3.5.8 Kasus Penggunaan Melihat Detail *Chart of Accounts*

Pada kasus penggunaan ini, sistem menampilkan halaman untuk melihat data *Chart of Accounts*. Pengguna dapat melihat data yang tersimpan pada tampilan yang tersedia. Tabel kasus penggunaan melihat data *Chart of Accounts* dapat dilihat pada Tabel 3.13 sedangkan gambar diagram aktivitas dapat dilihat pada Gambar 3.15.

Tabel 3.13. Kasus Penggunaan Melihat Data *Chart of Accounts*

Nama	**Melihat Detail *Chart of Accounts***
Nomor	UC-004
Deskripsi	Kasus penggunaan ini digunakan untuk melihat data *Chart of Accounts*
Tipe	Fungsional
Aktor	*Staff, Manager*
Kondisi Awal	Pengguna sudah login dan masuk ke dalam sistem
Kondisi Akhir	Sistem menampilkan halaman berupa data dari *chart of account* yang dipilih
Alur Normal	1. Sistem menampilkan halaman *Chart of Accounts* 2. Pengguna memilih *View* pada item tertentu 3. Sistem menampilkan halaman berupa data dari *chart of account* yang dipilih
Alur Alternatif	-



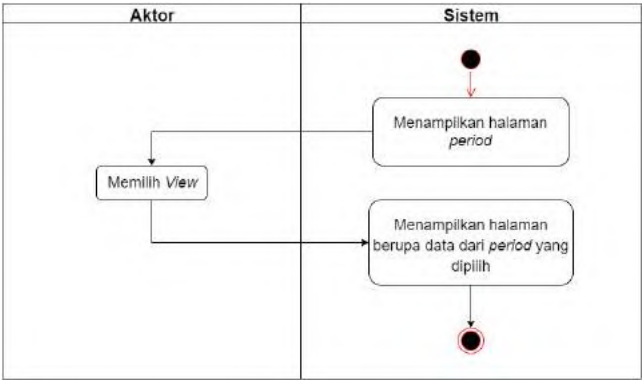
Gambar 3.15. Diagram Aktivitas Melihat Detail *Chart of Accounts*

3.5.9 Kasus Penggunaan Melihat Detail *Period*

Pada kasus penggunaan ini, sistem menampilkan halaman untuk melihat data *Period*. Pengguna dapat melihat data yang tersimpan pada tampilan yang tersedia. Tabel kasus penggunaan melihat data *Period* dapat dilihat pada tabel 3.14, sedangkan gambar diagram aktivitas dapat dilihat pada Gambar 3.16.

Tabel 3.14. Kasus Penggunaan Melihat Detail *Period*

Nama	**Melihat Data *Period***
Nomor	UC-006
Deskripsi	Kasus penggunaan ini digunakan untuk melihat data *Period*
Tipe	Fungsional
Aktor	*Staff, Manager*
Kondisi Awal	Pengguna sudah login dan masuk ke dalam sistem
Kondisi Akhir	Sistem menampilkan halaman berupa data dari *period* yang dipilih
Alur Normal	1. Sistem menampilkan halaman *Period* 2. Pengguna memilih *View* pada item tertentu 3. Sistem menampilkan halaman berupa data dari *period* yang dipilih
Alur Alternatif	-



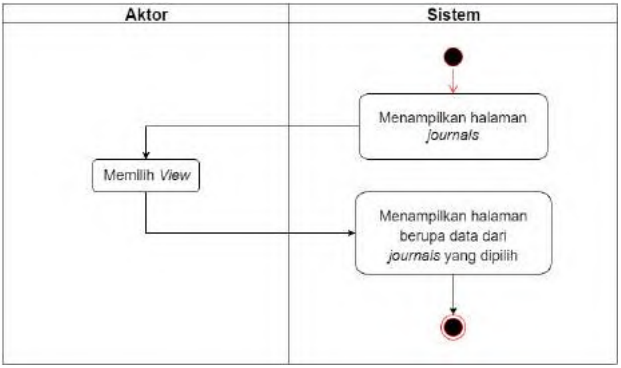
Gambar 3.16. Diagram Aktivitas Melihat Detail *Period*

3.5.10 Kasus Penggunaan Melihat Detail *Journals*

Pada kasus penggunaan ini, sistem menampilkan halaman untuk melihat data *Journals*. Pengguna dapat melihat data yang tersimpan pada tampilan yang tersedia. Tabel kasus penggunaan melihat data *Journals* dapat dilihat pada Tabel 3.15, sedangkan gambar diagram aktivitas dapat dilihat pada Gambar 3.17.

Tabel 3.15. Kasus Penggunaan Melihat Detail *Journals*

Nama	**Melihat Data *Journals***
Nomor	UC-007
Deskripsi	Kasus penggunaan ini digunakan untuk melihat data *Journals*
Tipe	Fungsional
Aktor	*Staff, Manager*
Kondisi Awal	Pengguna sudah login dan masuk ke dalam sistem
Kondisi Akhir	Sistem menampilkan halaman berupa data dari *journal* yang dipilih
Alur Normal	1. Sistem menampilkan halaman *Journal* 2. Pengguna memilih *View* pada item tertentu 3. Sistem menampilkan halaman berupa data dari *journal* yang dipilih
Alur Alternatif	-



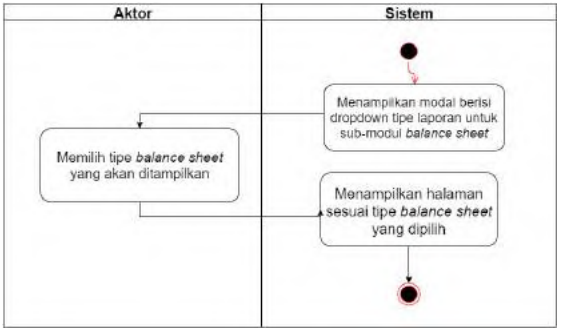
Gambar 3.17. Diagram Aktivitas Melihat Detail *Journals*

3.5.11 Kasus Penggunaan Melihat Laporan *Balance Sheet*

Pada kasus penggunaan ini, sistem menampilkan halaman untuk melihat laporan *balance sheet*. Pengguna dapat melihat laporan *balance sheet* pada tampilan yang tersedia. Tabel kasus penggunaan melihat laporan *balance sheet* dapat dilihat pada Tabel 3.16 sedangkan gambar diagram aktivitas dapat dilihat pada Gambar 3.18.

Tabel 3.16. Kasus Penggunaan Melihat Laporan *Balance Sheet*

Nama	**Melihat Laporan *Balance Sheet***
Nomor	UC-008
Deskripsi	Kasus penggunaan ini digunakan untuk melihat laporan *Balance Sheet*
Tipe	Fungsional
Aktor	*Staff, Manager*
Kondisi Awal	Pengguna sudah login dan masuk ke dalam sistem
Kondisi Akhir	Sistem menampilkan halaman berupa laporan *balance sheet* tertentu
Alur Normal	1. Sistem menampilkan modal untuk sub-modul *balance sheet* 2. Pengguna memilih tipe *balance sheet* yang akan ditampilkan pada modal yang muncul 3. Sistem menampilkan halaman sesuai tipe *balance sheet* yang dipilih pengguna
Alur Alternatif	-



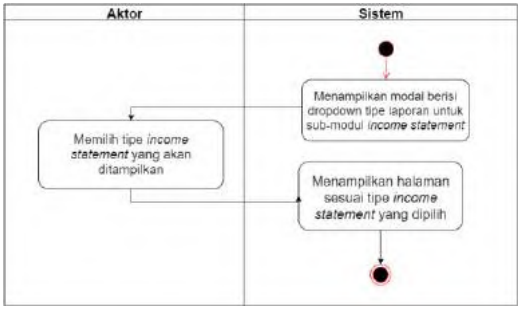
Gambar 3.18. Diagram Aktivitas Melihat Laporan *Balance Sheet*

3.5.12 Kasus Penggunaan Melihat Laporan *Income Statement*

Pada kasus penggunaan ini, sistem menampilkan halaman untuk melihat laporan *income statement*. Pengguna dapat melihat laporan pada tampilan yang tersedia. Tabel kasus penggunaan melihat laporan *income statement* dapat dilihat pada Tabel 3.17, sedangkan gambar diagram aktivitas dapat dilihat pada Gambar 3.19.

Tabel 3.17. Kasus Penggunaan Melihat Laporan *Income Statement*

Nama	**Melihat Laporan *Income Statement***
Nomor	UC-009
Deskripsi	Kasus penggunaan ini digunakan untuk melihat laporan *Income Statement*
Tipe	Fungsional
Aktor	*Staff*, *Manager*
Kondisi Awal	Pengguna sudah login dan masuk ke dalam sistem
Kondisi Akhir	Sistem menampilkan halaman berupa laporan *insome statement* tertentu
Alur Normal	1. Sistem menampilkan modal untuk sub-modul *income statement* 2. Pengguna memilih tipe *income statement* yang akan ditampilkan pada modal yang muncul 3. Sistem menampilkan halaman sesuai tipe *income statement* yang dipilih pengguna
Alur Alternatif	-



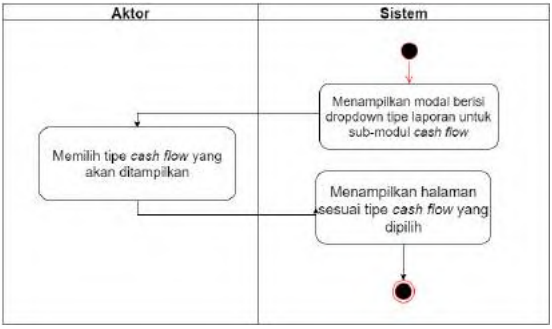
Gambar 3.19. Diagram Aktivitas Melihat Laporan *Income Statement*

3.5.13 Kasus Penggunaan Melihat Laporan *Cash Flow*

Pada kasus penggunaan ini, sistem menampilkan halaman untuk melihat laporan *financial statement* berupa *cash flow*. Pengguna dapat melihat laporan *cash flow* pada tampilan yang tersedia. Tabel kasus penggunaan melihat laporan *cash flow* dapat dilihat pada tabel 3.18, sedangkan gambar diagram aktivitas dapat dilihat pada Gambar 3.20.

Tabel 3.18. Kasus Penggunaan Melihat Laporan *Cash Flow*

Nama	**Melihat Laporan *Cash Flow***
Nomor	UC-013
Deskripsi	Kasus penggunaan ini digunakan untuk melihat laporan *Cash Flow*
Tipe	Fungsional
Aktor	*Staff, Manager*
Kondisi Awal	Pengguna sudah login dan masuk ke dalam sistem
Kondisi Akhir	Sistem menampilkan halaman berupa laporan *cash flow* tertentu
Alur Normal	1. Sistem menampilkan modal untuk sub-modul *cash flow* 2. Pengguna memilih tipe *cash flow* yang akan ditampilkan pada modal yang muncul 3. Sistem menampilkan halaman sesuai tipe *cash flow* yang dipilih pengguna
Alur Alternatif	-

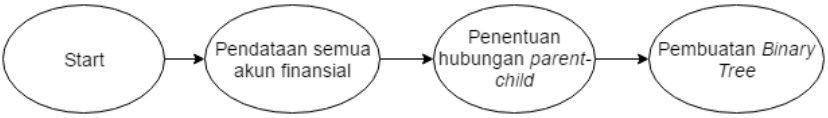


Gambar 3.20. Diagram Aktivitas Melihat Laporan *Cash Flow*

3.6 Account Charting

Account Charting atau pembuatan *chart of accounts* adalah hal yang sangat mendasar dari akuntansi, karena akun pada *chart of accounts* tidak hanya digunakan untuk pembuatan jurnal tapi yang juga akan menentukan laporan keuangan nantinya.

Dalam pembuatan *chart of accounts*, terdapat beberapa tipe cara pembuatannya. Pada buku ini akan diterangkan bagaimana cara membuat akun menggunakan algoritma *Binary Tree*. Gambar 3.21. meringkaskan langkah-langkah dalam pembuatannya.



Gambar 3.21. Proses pembuatan akun

3.6.1 Pendataan Semua Akun Finansial

Pada tahap ini, semua adalah kebijakan dari perusahaan. Pendataan juga termasuk pemecahan akun yang masih umum ke dalam akun yang lebih detil. Seperti contoh, akun "*Machine*" masih dapat dipecah lagi menjadi misal "*Welding Machine", "Spray Painting Machine",* dsb. Penentuan akun tidak bisa dilakukan oleh akuntan sendiri, ditambah apabila terkait dengan aset yang dibuat oleh *Asset Management*, maka harus melapor ke akuntan dahulu untuk dibuatkan akunnya. Contoh hasil dari tahap ini ditampilkan pada Tabel 3.19.

Tabel 3.19. Contoh Pendataan Akun Finansial

Account Name
Machine
Property Plant and Equipment
Welding Machine
Molding Machine

Vehicle
Truck
Equipment
Forklift

Pada Tabel 3.19, ditampilkan bahwa data-data akun finansial yang baru disusun tidak harus tersusun berurutan sesuai dengan pecahannya.

3.6.2 Penentuan Hubungan *Parent-Child*

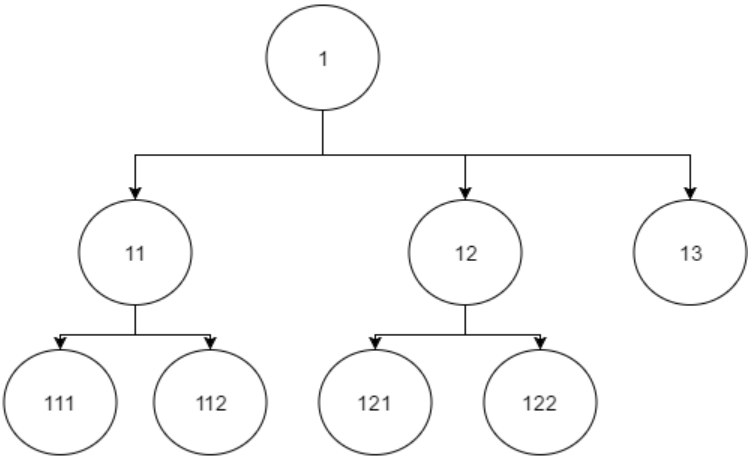
Pada tahap ini, akun-akun yang telah didata pada tahap sebelumnya, diurutkan berdasarkan hubungan *parent-child*-nya. Pengurutan juga disertai dengan pemberian kode pada masing-masing akun. Contoh hasil dari tahap ini ditampilkan pada Tabel 3.20.

Tabel 3.20. Contoh Penentuan Hubungan *Parent-Child*

Account Code	Account Name
1	*Property Plant and Equipment*
11	*Machine*
111	*Welding Machine*
112	*Molding Machine*
12	*Vehicle*
121	*Truck*
122	*Forklift*
13	*Equipment*

3.6.3 Pembuatan *Binary Tree*

Dari 2 tahap sebelumnya sehingga didapatkan kode dari masing-masing akun. Lalu dibuat *binary tree* dari akun-akun tersebut sesuai dengan relasi *parent-child* masing-masing akun. Contoh *binary tree* yang dihasilkan dari tahap ini ditampilkan pada Gambar 3.22.



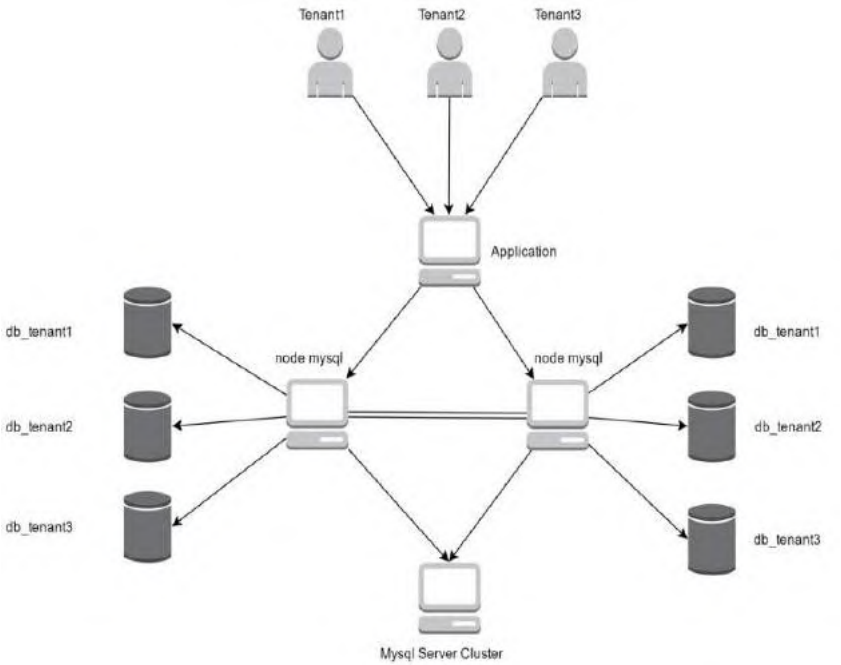
Gambar 3.22. Contoh *Pembuatan Binary Tree*

3.7 Perancangan Sistem

Tahap ini meliputi perancangan basis data, tampilan antarmuka, dan perancangan alur proses penggunaan sistem pada modul *Accounting* yang diharapkan dapat memenuhi tujuan dari pengembangan aplikasi ini.

3.7.1 Perancangan *Multitenancy*

Pada bagian ini akan dijelaskan mengenai perancangan *multitenancy* yang terdapat pada sistem ERP 2016. Secara rinci mengenai *multitenancy* dijabarkan pada Gambar 3.23.

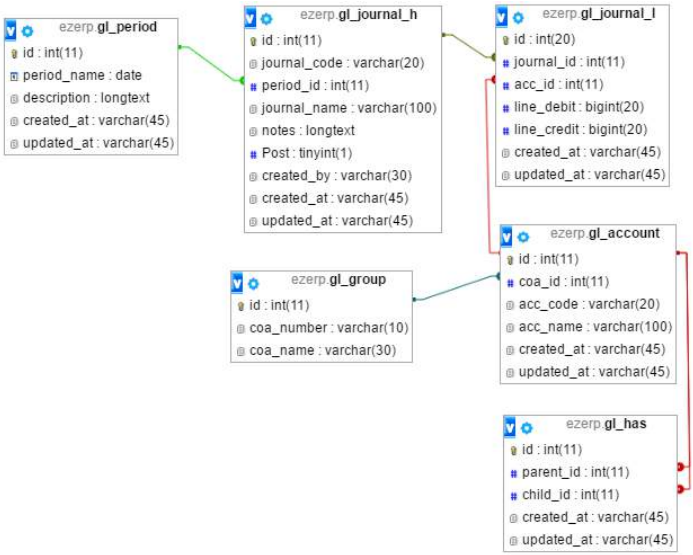


Gambar 3.23. Perancangan *Multi-tenancy*

Gambar 3.23 menunjukkan alur *multitenancy* yang terjadi pada sistem ERP. Pembagian node MySQL yang telah dijelaskan bada sub-bab 3.2.1 yaitu dengan cara melakukan *clustering* pada computer server menjadi 2 basis data yang berbeda. Basis data terdistribusi merupakan penunjang dari *multitenancy*, yang berfungsi untuk mengatur *session* setiap perusahaan yang akan mengakses sistem ERP.

3.7.2 Perancangan Basis Data

Pada Subbab ini akan dijelaskan bagaimana rancangan basis data yang digunakan pada sistem. Basis data pada sistem ini menggunakan sistem manajemen basis data *MySQL Cluster*. *Physical Data Model* (PDM) dari basis data sistem ini ditunjukkan pada Gambar 3.24. Perancangan PDM pada Gambar 3.32 dijelaskan pada Tabel 3.21.



Gambar 3.24. *Physical Data Model* Modul *General Ledger*

Tabel 3.21. Keterangan *Physical Data Model* Modul *General Ledger*

No.	Nama Tabel	Atribut	Keterangan
1	gl_period	id : ***integer***	Tabel untuk menyimpan data *period*
		period_name : ***date***	
		description : ***longtext***	
		created_at : ***varchar***	
		updated_at : ***varchar***	
2	gl_journal_h	id : ***integer***	Tabel untuk menyimpan data yang berhubungan dengan *journal*
		journal_code : ***varchar***	
		period_id : ***integer***	
		journal_name : ***varchar***	
		notes : ***longtext***	
		Post : ***boolean***	
		created_by : ***varchar***	
		created_at : ***varchar***	
		updated_at : ***varchar***	
3	gl_journal_l	id : ***integer***	
		journal_id : ***integer***	

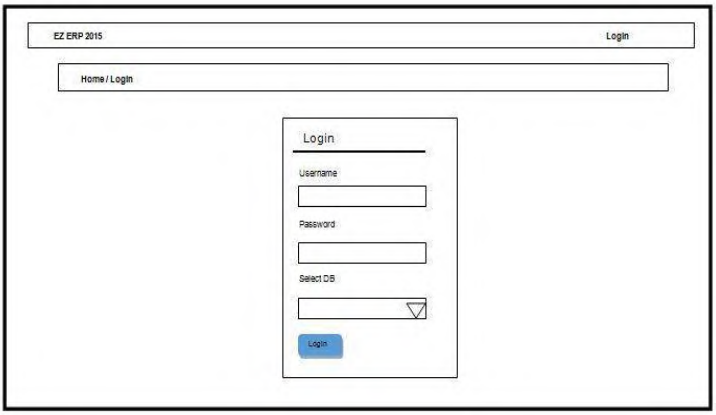
No.	Nama Tabel	Atribut	Keterangan
		acc_id : ***integer***	Tabel untuk menyimpan data isi dari masing-masing *journal*
		line_debit : ***longint***	
		line_credit : ***longint***	
		created_at : ***varchar***	
		updated_at : ***varchar***	
4	gl_group	id : ***integer***	Tabel master yang berisi tipe-tipe akun yang dapat dipilih
		coa_number : ***integer***	
		coa_name : ***varchar***	
5	gl_account	id : ***integer***	Tabel untuk menyimpan data-data akun dari *Chart of Accounts*
		coa_id : ***integer***	
		acc_code : ***varchar***	
		acc_name : ***varchar***	
		created_at : ***varchar***	
		updated_at : ***varchar***	
6	gl_has	id : ***integer***	Tabel untuk menyimpan relasi antar akun
		parent_id : ***integer***	
		child_id : ***integer***	
		created_at : ***varchar***	
		updated_at : ***varchar***	

3.7.3 Perancangan *Role Based Access Control* (RBAC)

Pada bagian ini akan dijelaskan mengenai perancangan RBAC yang terdapat pada sistem ERP 2016. Perincian mengenai RBAC dijabarkan sebagai berikut

3.7.3.1 Perancangan Antarmuka *Login*

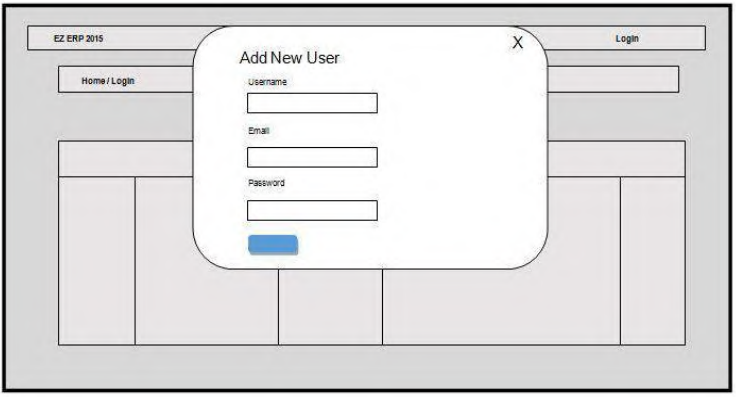
Rancangan antarmuka login dibagi menjadi 3 (tiga) bagian, di bagian paling atas terdapat header yang berisi logo aplikasi dan tombol login, di bagian atas terdapat navigasi (*breadcrumb*s) untuk mengetahui user sedang berada pada halaman tententu, dan di bagian tengah terdapat form *login* yang memiliki *username*, *password*, dan pemilihan basis data yang digunakan sebagai input. Perancangan antarmuka *login* ditunjukkan pada Gambar 3.25.



Gambar 3.25. Perancangan Antarmuka *Login*

3.7.3.2 Perancangan Antarmuka Menambahkan *User* Baru

Rancangan antarmuka menambahkan *user* baru ditampilkan dalam sebuah kotak dialog yang berisi *username*, *email*, dan *password* yang digunakan sebagai input data untuk menambahkan user baru. Perancangan antarmuka menambahkan user baru ditunjukkan pada Gambar 3.26.



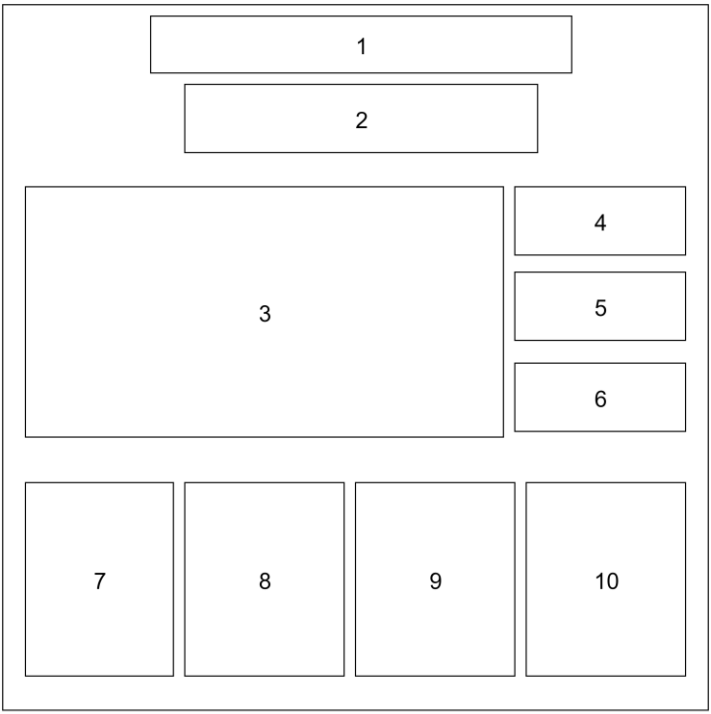
Gambar 3.26. Perancangan Antarmuka Menambahkan *User* Baru

3.7.4 Perancangan Antarmuka Sistem

Pada sub-bab ini membahas perancangan antarmuka yang akan digunakan dalam aplikasi ERP khususnya modul *general ledger*. Ilustrasi perancangan ditunjukkan sebagai gambar disertai keterangan objek-objek yang ada di dalamnya.

3.7.4.1 Perancangan Antarmuka Halaman *Dashboard*

Rancangan antarmuka halaman awal pada modul *general ledger* ditampilkan pada Gambar 3.27.



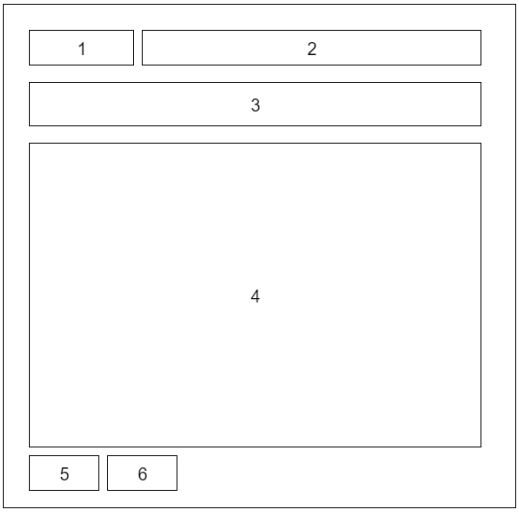
Gambar 3.27. Perancangan Antarmuka Halaman *Dashboard*

Penjelasan angka-angka yang tertera pada Gambar 3.27. adalah sebagai berikut :

1. “General Ledger”
2. Submodul-submodul yang ada pada modul *General Ledger*
3. Grafik perkembangan keuntungan yang didapat perusahaan
4. Uang yang tersisa di perusahaan
5. Keuntungan yang didapat hari ini
6. Pengingat apabila ada jurnal yang belum dipost
7. *Financial Statement Analysis : Liquidity Ratio*
8. *Financial Statement Analysis : Activity Analysis Ratio*
9. *Return of Assets (ROA)*
10. *Profit Margin*

3.7.4.2 Perancangan Antarmuka Halaman *Chart of Accounts*

Halaman berikut merupakan halaman *chart of accounts.* Halaman ini berguna untuk memasukkan data *chart of accounts* yang baru ke dalam sistem. Rancangan antarmuka halaman ini adalah sebagai berikut.



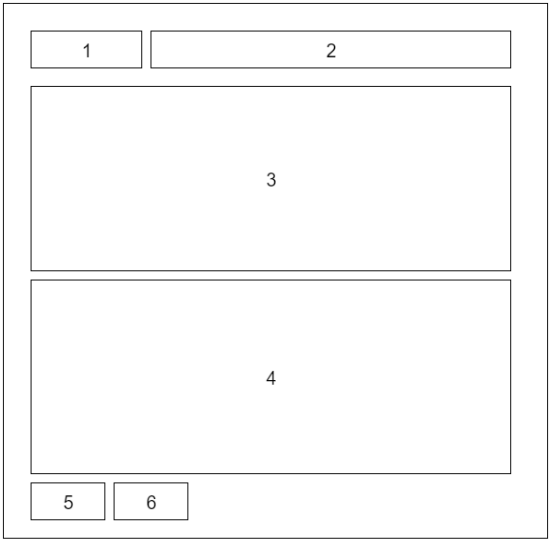
Gambar 3.28. Perancangan Antarmuka Halaman *Chart of Accounts*

Penjelasan masing-masing angka dari gambar 3.28. adalah sebagai berikut:

1. *Input* untuk *Account Code*
2. *Input* untuk *Account Name*
3. Memilih tipe akun dari akun yang diinput (*Assets, Liabilities, Equities, Revenues, Expenses*)
4. Tabel yang bisa digunakan untuk memasukkan data anak-anak dari akun yang diinputkan. Berisi form yang sama (berisi form dari 1-3).
5. Tombol *Create*
6. Tombol *Cancel*

3.7.4.3 Perancangan Antarmuka Halaman *Journal*

Halaman ini digunakan untuk membuat jurnal keuangan bagi perusahaan. Perancangan dari antarmuka ini tertera pada Gambar 3.29.



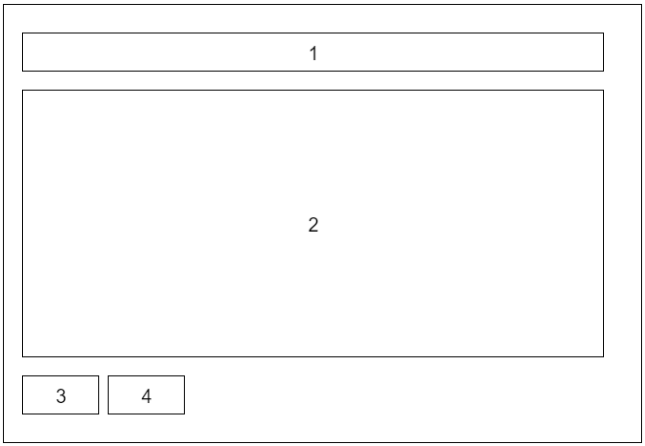
Gambar 3.29. Perancangan Antarmuka Halaman *Journal*

Penjelasan masing-masing angka pada Gambar 3.29. adalah sebagai berikut:

1. *Input* untuk *Journal Code*
2. *Input* untuk *Journal Name*
3. *Input* untuk *notes* masing-masing jurnal
4. Tabel untuk transaksi yang terjadi di jurnal tersebut
5. Tombol *Create*
6. Tombol *Cancel*

3.7.4.4 Perancangan Antarmuka Halaman *Period*

Halaman ini digunakan untuk membuat periode akuntansi yang digunakan untuk merekam jurnal-jurnal yang terjadi pada periode tersebut. Perancangan antarmuka dari halaman ini tertera pada Gambar 3.30.



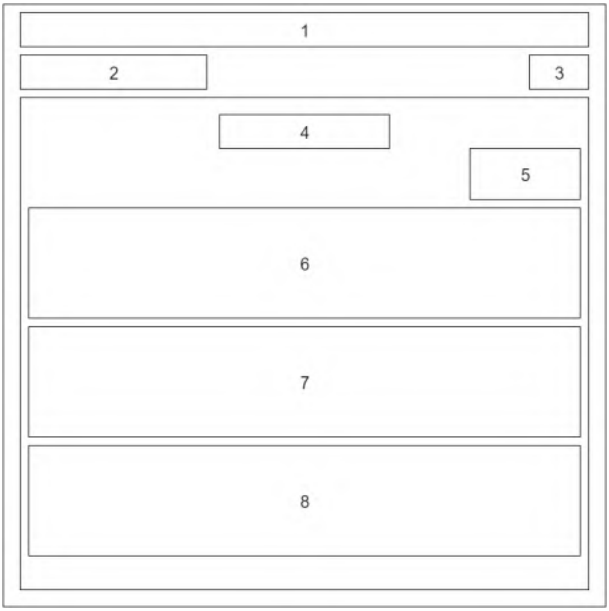
Gambar 3.30. Perancangan Antarmuka Halaman *Period*

Berikut penjelasan masing-masing angka pada Gambar 3.30.:

1. *Input* untuk *Period Name (Date)*
2. *Input* untuk *Period Description*
3. Tombol *Create*
4. Tombol *Cancel*

3.7.4.5 Perancangan Antarmuka Halaman *Balance Sheet*

Halaman ini digunakan untuk menampilkan laporan dari transaksi-transaksi finansial yang terjadi di perusahaan. Perancangan antarmuka halaman *balance sheet* adalah sebagai berikut:



Gambar 3.31. Perancangan Antarmuka Halaman *Balance Sheet*

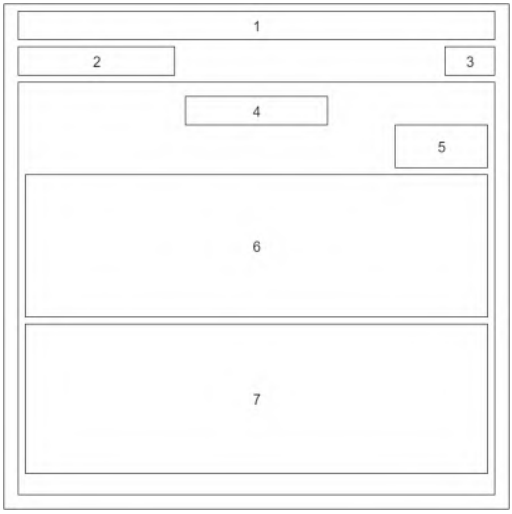
Penjelasan masing-masing angka pada Gambar 3.31. adalah sebagai berikut:

1. *Dropdown* antara tahunan, bulanan, atau harian.
2. *Dropdown* tergantung pilihan pada nomor 1.
   a. Tahunan : *Dropdown* tahun yang telah terjadi transaksi
   b. Bulanan : *Dropdown* bulan serta tahun yang telah terjadi transaksi
   c. Harian : *Dropdown datepicker*

3. Mencetak halaman laporan menjadi PDF
4. Header laporan (*Annual, Monthly, Daily*)
5. Deskripsi lebih lanjut mengenai laporan (tanggal terakhir transaksi, dsb.)
6. Semua akun *Assets* yang telah terjadi transaksi
7. Semua akun *Liabilities* yang telah terjadi transaksi
8. Semua akun *Equities* yang telah terjadi transaksi

3.7.4.6 Perancangan Antarmuka Halaman *Income Statement*

Halaman ini digunakan untuk menampilkan laporan dari transaksi-transaksi finansial yang berhubungan dengan biaya operasional yang terjadi di perusahaan. Perancangan antarmuka halaman *income statement* adalah sebagai berikut:



Gambar 3.32. Perancangan antarmuka Halaman *Income Statement*

Penjelasan masing-masing angka pada Gambar 3.32. adalah sebagai berikut:

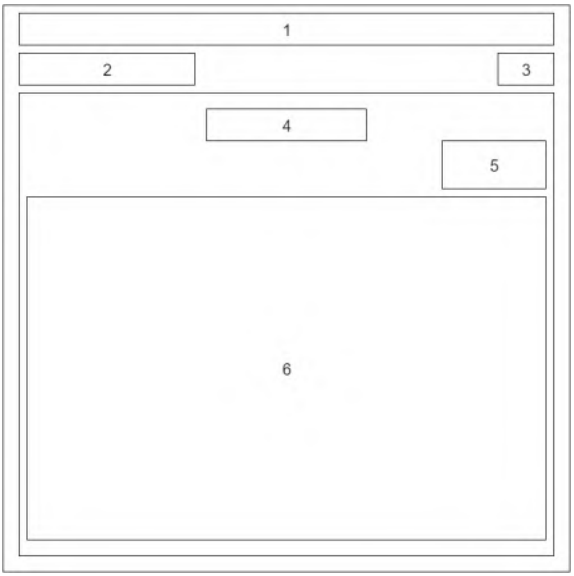
1. *Dropdown* antara tahunan, bulanan, atau harian.
2. *Dropdown* tergantung pilihan pada nomor 1.

a. Tahunan : *Dropdown* tahun yang telah terjadi transaksi
b. Bulanan : *Dropdown* bulan serta tahun yang telah terjadi transaksi
c. Harian : *Dropdown datepicker*

3. Mencetak halaman laporan menjadi PDF
4. Header laporan (Annual, Monthly, Daily)
5. Deskripsi lebih lanjut mengenai laporan (tanggal terakhir transaksi, dsb.)
6. Semua akun *Revenues* yang telah terjadi transaksi
7. Semua akun *Expenses* yang telah terjadi transaksi

3.7.4.7 Perancangan Antarmuka Halaman *Cash Flow*

Halaman ini digunakan untuk menampilkan laporan dari aliran uang yang terjadi di perusahaan, seperti pembelian *raw materials*, hasil penjualan, dsb. Perancangan antarmuka halaman *cash flow* adalah sebagai berikut:



Gambar 3.33. Perancangan Antarmuka Halaman *Cash Flow*

Penjelasan masing-masing angka pada Gambar 3.33. adalah sebagai berikut:

1. *Dropdown* antara tahunan, bulanan, atau harian.
2. *Dropdown* tergantung pilihan pada nomor 1.
   a. Tahunan : *Dropdown* tahun yang telah terjadi transaksi
   b. Bulanan : *Dropdown* bulan serta tahun yang telah terjadi transaksi
   c. Harian : *Dropdown datepicker*
3. Mencetak halaman laporan menjadi PDF
4. Header laporan (*Annual, Monthly, Daily*)
5. Deskripsi lebih lanjut mengenai laporan (tanggal terakhir transaksi, dsb.)
6. Semua akun yang berhubungan dengan *cash* yang telah terjadi transaksi

BAB IV
IMPLEMENTASI SISTEM

Bab ini membahas implementasi dari perancangan sistem ERP 2015. Di dalamnya mencakup penjelasan lingkungan pengembangan sistem serta proses implementasi *distributed database, RBAC, multi-tenancy*, dan antarmuka pengguna.

4.1 Lingkungan Pengembangan Sistem

Lingkungan pengembangan sistem yang digunakan untuk mengembangkan tugas akhir ini dilakukan pada lingkungan dan kakas sebagai berikut:

1. Basis data yang digunakan pada *server* adalah SQL Server 2012 R2.
2. PC untuk *server* menggunakan AMD A6-3400M APU with Radeon HD Graphics @1.40Ghz, RAM 4GB dengan Sistem Operasi Windows 7 Ultimate.
3. Draw.io untuk pembuatan diagram dan desain antarmuka, Sublime Text sebagai teks editor, MySQL Workbench untuk pembuatan PDM.
4. Google Chrome sebagai antarmuka untuk pengujian aplikasi klien.

4.2 Implementasi *Distributed Database*

Pada bagian ini akan dijelaskan mengenai implementasi *distributed database* yang terdapat pada sistem ERP 2015. Secara rinci mengenai implementasi *distributed database* dijabarkan sebagai berikut:

4.2.1 Instalasi Data dan SQL node pada node 1 dan node 2

Pada implementasi instalasi dan sql node pada *node* 1 dan *node* 2, dilakukan langkah-langkah sebagai berikut:

1. Membuat grup MySQL pengguna baru, kemudian menambah user MySQL. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.1.

```
shell> groupadd mysql
shell> useradd-g mysql mysql
```

Kode Sumber 4.1. Membuat grup MySQL pengguna baru dan menambah user MySQL

2. Mengubah lokasi ke dalam direktori yang berisi file yang telah didownload, kemudian mengubah arsip dan menciptakan symlink ke dalam direktori MySQL yang bernama "mysql". Hal yang perlu diperhatikan adalah, file yang sebenarnya dan nama direktori bervariasi sesuai dengan jumlah cluster versi MySQL. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.2.

```
shell> cd/usr/local
shell>/usr/local$ tar xzvf mysql-cluster-gpl-7.1.34-
      linux-x86_64-glibc23
shell>ln-s/usr/local/mysql-cluster-gpl-7.1.34-linux-
      x86-glibc23/usr/local/mysql
shell> export PATH = $ PATH:/usr/local/mysql/bin
shell>    echo    "export    PATH=\$PATH:/usr/local/
      mysql/bin"›› /etc/bash.bashrc
```

Kode Sumber 4.2. Mengubah lokasi Direktori, Mengubah Arsip, dan Menciptakan *Symlink*

3. Mengubah lokasi ke dalam direktori MySQL dan menjalankan *script* untuk menciptakan *database system*. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.3.

```
shell> cd mysql
shell>. / scripts / mysql_install_db-user = mysql
```

Kode Sumber 4.3. Mengubah Lokasi Direktori

4. Mengatur izin yang diperlukan oleh server MySQL. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.4.

```
shell> chown-R root.
shell> chown-R mysql data
shell> chgrp-R mysql.
```

Kode Sumber 4.4. Mengatur Perizinan *Server* MySQL

5. Menyalin *script startup MySQL* ke direktori yang sesuai, mengubah menjadi *executable*, dan memulai ketika sistem beroperasi. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.5.

```
shell> cp support-files/mysql.server / etc / init.d /
mysql
shell> chmod + x / etc / init.d / mysql
shell> update-rc.d mysql defaults
```

Kode Sumber 4.5. Menyalin Script Startup MySQL

4.2.2 Pemasangan node manajamen pada node 3

Pemasangan *node* manajemen memerlukan manajemen *server* MySQL Cluster (ndb_mgmd), diasumsikan bahwa mysql-cluster-gpl-7.1.5-linux-i686-glibc23.tar.gz telah ditempatkan di /var / tmp. Untuk memasang ndb_mgmd dan ndb_mgm pada host *Cluster*, sistem sebagai *root* melakukan langkah-langkah sebagai berikut:

1. Mengubah lokasi ke dalam direktori / var / tmp direktori, dan mengekstrak ndb_mgm dan ndb_mgmd dari arsip ke direktori yang sesuai seperti / usr / local / bin. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.6.

```
shell> cd/usr/local
shell> tar-zxvf mysql-cluster-gpl-7.1.34-linux-x86-
glibc23.tar.gz
shell> cd / usr/local/mysql-cluster-gpl-7.1.34-
linux-x86-glibc235
shell> cp bin / ndb_mgm */usr/local/bin
```

Kode Sumber 4.6. Mengubah Lokasi Direktori

2. Mengubah lokasi ke dalam direktori tempat *file* disalin, kemudian dieksekusi. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.7.

```
shell> cd/usr/local/bin
shell> chmod +x ndb_mgm*
```

Kode Sumber 4.7. Mengubah Lokasi Direktori

4.2.3 Konfigurasi Manajemen Node

Konfigurasi pada manajemen *node* dilakukan dengan langkah-langkah sebagai berikut:

1. Membuat direktori tempat *file* konfigurasi ditemukan kemudian membuat *file* itu sendiri. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.8.

```
shell> mkdir / var / lib / mysql-cluster
shell> cd / var / lib / mysql-cluster
shell> vi config.ini
```

Kode Sumber 4.8. Membuat Direktori

2. Mengatur *file* "config.ini". Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.9.

```
    NoOfReplicas=2
    DataMemory=10G
    IndexMemory=2G
    MaxNoOfAttributes=10000
    MaxNumberOfTables=2500
    MaxOfOrderedIndexes=4086
    MaxOfConcurrentOperations=250000
    MaxOfConcurrentOperations=250000
    [tcp default]
    [ndb_mgmd]
    hostname=10.151.64.182
    datadir=/var/lib/mysql-cluster
    [ndbd]
    hostname=10.151.64.182
    datadir=/usr/local/mysql/data
    [ndbd]
    hostname=10.151.64.203
    datadir=/usr/local/mysql/data
    [mysqld]
    MaxNoOfAttributes=10000
```

```
hostname=10.151.64.182
[mysqld]
MaxNoOfAttributes=10000
hostname=10.151.64.203
```

Kode Sumber 4.9. Mengatur file "config.ini"

4.2.4 Konfigurasi data dan SQL node

Konfigurasi data dan SQL Node dilakukan dengan cara mengedit *file* my.cnf pada direktori /etc/. Untuk setiap data *node* dan SQL node yang diatur pada my.cnf. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.10.

```
[client]
port=3306
socket=/tmp/mysql.sock
[mysqld]
port=3306
socket-/tmp/mysql.sock
ndbcluster
ndb-connectstring=10.151.64.181
[mysql_cluster]
ndb-connectstring=10.151.64.181
```

Kode Sumber 4.10. Data dan SQL Node

4.2.5 Memulai MySQL Cluster

Setiap proses *node cluster* harus dimulai secara terpisah. Manajemen *node* harus dimulai terlebih dahulu, kemudian *node* data. Pada setiap *node* SQL dilakukan langkah sebagai berikut :

1. Pada *node* 03 (host manajemen), untuk memulai proses manajemen *node* dari *shell* sistem dilakukan perintah berikut. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.11.

```
shell> ndb_mgmd -f /var/lib/mysql-
cluster/config.ini-configdir=/var/lib/mysql-
cluster/
```

Kode Sumber 4.11. Memulai Proses Manajemen Node

2. Jalankan perintah untuk memulai ndbd dan proses MySQL server pada masing-masing Data/host SQL. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.12.

```
shell> / usr / local / mysql / bin / ndbd
```

Kode Sumber 4.12. Memulai proses ndbd dan proses mysql server.

3. Mengaktifkan MySQL pada data *node*. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.13.

```
shell> / etc / init.d / mysql start
```

Kode Sumber 4.13. Mengaktifkan MySQL

4.3 Implementasi RBAC (*Role-Based Access Control*)

Pada bagian ini akan dijelaskan mengenai implementasi *RBAC* yang terdapat pada sistem ERP 2016. Secara rinci mengenai implementasi *RBAC* dijabarkan sebagai berikut:

4.3.1 Membuat Tabel Pengguna

Pembuatan tabel pengguna dilakukan dengan menjalankan perintah "*yii migrate*" pada folder aplikasi. Apabila proses migrasi berhasil dilakukan, maka akan muncul keterangan "*migrate successfully*" dan akan muncul tabel *user* di dalam basis data.

4.3.2 Membuat 4 Tabel Autentifikasi RBAC dan Tabel Pengguna

Pada tahap ini dibutuhkan 4 tabel autentifikasi yang terdiri dari:

1. Tabel Item, tabel yang menyimpan daftar otorisasi, standar nama tabel adalah auth_item.
2. Tabel Child, tabel yang menyimpan hirarki daftar otoristas, standar nama tabel adalah auth_item_child.

3. Tabel Assigment, tabel yang menyimpan penetapan user untuk daftar otorisasi, standar nama tabel adalah auth_assignment.
4. Tabel Rule, tabel yang menyimpan aturan-aturan autentifikasi, standar nama tabel adalah auth_rule.

Cara untuk membuat tabel-tabel tersebut yaitu dengan menjalankan perintah yang ditunjukkan pada Kode Sumber 4.14.

```
Yii migrate --
migrationPath=@yii/rbac/migrations
```

Kode Sumber 4.14. Generate Tabel Autentifikasi

Jika berhasil yang maka akan terbuat 4 tabel pada *database*, yaitu tabel ***auth_assigment***, tabel ***auth_item***, tabel ***auth_item_child*** dan tabel ***auth_rule***.

4.3.3 Membuat Modul Admin dan Konfigurasi Autentikasi

Proses ini bertujuan untuk meletakkan konfigurasi pengguna dan masing-masing model dari 4 tabel autentifikasi. Kemudian dilakukan konfigurasi pada folder config *file* `web.php`. Terdapat 3 konfigurasi yaitu:

1. *Admin* : digunakan sebagai akses ke modul *admin*
2. *Auth Manager* : sebagai autentifikasi di yii2 dan mengatur role default sebagai *guest*
3. *Session Time Out* : mengatur durasi *time out session* selama 5 menit atau 300 detik.

Masing –masing konfigurasi tersebut ditunjukkan pada Kode Sumber 4.15.

```
$config = [
     //…
     'modules' => [
          'admin' => [
               'class'=>'app\modules\admin\AdminModule,
```

```
            ],
      ],
      'components' => [
            'authManager' => [
                  'class' => 'yii\rbac\DbManager',
                  'defaultRoles' => ['guest'],
            ],
            'session' => [
                  'timeout' => 300,
            ],
            //…
      ],
      //…
];
```

Kode Sumber 4.15. Pembuatan Modul Admin

4.3.4 Membuat Model Tabel Autentifikasi, *Controller*, dan *View* Pengguna

Pada tahap ini dilakukan *generate* kelas model dari masing-masing tabel autentifikasi pada modul *admin* yang diperlukan pada tahap sebelumnya. Kemudian ditambahkan generate kelas *controller* dan *view* pada tabel pengguna.Proses generate ini menggunakan *yii generator* yang telah disediakan oleh *framework* yii.

4.3.5 Menambahkan Kode pada *usercontroller*

Pada modul admin, file *usercontroller.php* ditambahkan kode fungsi untuk semua tabel autentifikasi. Masing-masing fungsi ditunjukkan pada Kode Sumber 4.16, Kode Sumber 4.17, Kode Sumber 4.18.

```
public function actionAuthItem()
{
   $auth = Yii::$app->authManager;

  // menambahkan akses sebagai admin ke tabel auth_item
  $admin = $auth->createPermission('admin');
  $admin->description = 'Allow user to access all page';
  $auth->add($admin);
```

```
    // menambahkan akses sebagai asset management manajer ke
    tabel auth_item
    $am_manager = $auth->createPermission('am-manager');
    $am_manager->description = 'Allow user as Asset Management
    Manager';
    $auth->add($am_manager);

    // menambahkan akses sebagai asset management staff ke
    tabel auth_item
    $am_staff = $auth->createPermission('am-staff');
    $am_staff->description = 'Allow user as Asset Management
    Staff';
    $auth->add($am_staff);
```

Kode Sumber 4.16. Kode Fungsi Tabel AuthItem

```
public function actionItemChild(){
     $auth = Yii::$app->authManager;

     //admin dapat mengakses semua daftar izin akses
     $am_manager = $auth->createPermission('am-manager');
     $am_staff = $auth->createPermission('am-staff');
     $ap_manager = $auth->createPermission('ap-manager');
     $ap_staff = $auth->createPermission('ap-staff');

     $admin = $auth->createRole('admin');
     $auth->add($admin);
     $auth->addChild($admin, $am_manager);
     $auth->addChild($admin, $am_staff);
     $auth->addChild($admin, $ap_manager);
     $auth->addChild($admin, $ap_staff);
}
```

Kode Sumber 4.17. Kode Fungsi Tabel ItemChild

```
public function actionAuthAssignment(){
     $auth = Yii::$app->authManager;

     $admin = $auth->createRole('admin');

     $auth->assign($admin, 1);
 }
```

Kode Sumber 4.18. Kode Fungsi Tabel AuthAssignment

4.4 Implementasi Multitenancy

Pada bagian ini akan dijelaskan mengenai implementasi *RBAC* yang terdapat pada sistem ERP 2016. Secara rinci mengenai implementasi *RBAC* dijabarkan sebagai berikut:

4.4.1 Membuat Halaman Muka Tenant

Pada implementasi membuat halaman *tenant* ini dilakukan pembuatan halaman *tenant* secara sederhana, kemudian ditambahkan pembuatan *database* untuk setiap tenant. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.19.

```
// mendapatkan nilai yang dimasukkan dari view
   $host = "10.151.64.182";
   $tenant = Yii::$app->request->post('tenant');
   $database = Yii::$app->request->post('database');

// membuka koneksi pada untuk memasukkan data tenant
$koneksidb = mysqli_connect($host, "root", "", "multitenant");

if ($koneksidb->connect_error) {
    die("Connection failed: " . $koneksidb->connect_error);
 }
// query untuk memasukkan data tenant
$insert = "insert into tenant
           (tenant,database_name,created_at) values
           ('".$tenant."','".$database."',NOW())";

 if($tenant != "" && $database != ""){
      $koneksidb->query($insert);
 }
// memutuskan koneksi ke server
                  $koneksidb->close();
```

Kode Sumber 4.19. Pembuatan Halaman Muka Tenant

4.4.2 Menambahkan Basis Data untuk *Tenant* Baru

Pada tahap ini dilakukan penambahan *database* untuk *tenant* baru dengan cara melakukan konfigurasi pada server *node* MySQL. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.20

```
'mysql –u root –p;' lalu tekan enter
'create database <nama database>;' lalu tekan enter
'use <nama database>;' lalu tekan enter
 e /tmp/mysql-dump/final-db.sql
```

Kode Sumber 4.20. Pembuatan Database Tenant Baru

4.4.3 *Login* Tenant

Setelah proses pembuatan *database* dan replikasi pada tahap 4.4.2. selesai, maka tenant melakukan *login* dengan memilih salah satu *database*, nama *database* yang dipilih tersebut disimpan dalam session dan akan digunakan untuk koneksi yang akan dibuat. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.21.

```
$host = '10.151.64.182';  // node mysql
$port = 3306;
$waitTimeoutInSeconds = 1;
$rc =
     @fsockopen($host,$port,$errCode,$errStr,$waitTimeoutInSeconds);
if(is_resource($rc)){
   $_SESSION['dbserver_ip'] = "10.151.64.182"; // node mysql
   // penyimpanan nama database sebagai session untuk multitenancy
   $dbname = isset($_SESSION['database_name']) ?
     $_SESSION['database_name'] : 'test';
} else {
   $_SESSION['dbserver_ip'] = "10.151.64.203";  // node mysql
}
    $connection = [
        'class' => 'yii\db\Connection',
        'dsn' =>
     'mysql:host='.$_SESSION['dbserver_ip'].';dbname='.$dbname.'',
        'username' => 'root',
        'password' => '',
        'charset' => 'utf8'
    ];
return $connection;
```

Kode Sumber 4.21. Login Tenant

4.5 Implementasi Program pada Modul *General Ledger*

Pada bagian ini akan dijelaskan mengenai implementasi yang terdapat pada modul *General Ledger* yang terbagi menjadi tampilan halaman utama atau *dashboard* dan beberapa sub-modul, antara lain:

Period, Journals, Account Balance, Balance Sheet, Income Statement, dan *Cash Flow*.

4.5.1 Halaman Utama Modul General Ledger

Halaman utama modul *General Ledger* menampilkan informasi pencatatan perkembangan keuntungan finansial serta analisis laporan keuangan secara grafis. Pengaturan tampilan tersebut diatur di kelas *DefaultController* dalam fungsi *actionIndex()* . Kode tampilan yang dimaksud ditunjukkan oleh kode sumber B.1.

4.5.2 Melihat Data Submodul Modul General Ledger

Pada implementasi melihat daftar *Chart of Accounts*, *Journals* ,dan *Period*, sistem menampilkan seluruh daftar akun yang telah ditambahkan pada basis data *chart of accounts*. Pengaturan tampilan ini diatur pada fungsi *actionIndex()* Kode Sumber untuk menampilkan seluruh transaksi tersebut ditunjukkan pada kode sumber B.2.

4.5.3 Menambah Data Submodul Modul General Ledger

Pada implementasi menambah *Chart of Accounts*, *Journals*, dan *Period*, sistem menambahkan transaksi baru ke dalam *Chart of Accounts*. Pengaturan manambah transaksi baru ini diatur pada fungsi *actionCreate()*. Kode Sumber untuk pembuatan data ditunjukkan pada kode sumber B.4.

4.5.4 Menyunting Data Submodul Modul General Ledger

Pada implementasi menyunting *Chart of Accounts*, *Journals*, dan *Period*, sistem memperbarui data yang ada pada *Chart of Accounts*. Pengaturan menyunting transaksi ini diatur pada fungsi *actionUpdate()*. Kode Sumber untuk menyunting data ditunjukkan pada kode sumber B.5.

4.5.5 Menghapus Data Submodul Modul General Ledger

Pada implementasi menghapus *Chart of Accounts*, *Journals,* dan *Period,* pengaturan penghapusan data ini diatur pada fungsi *actionDelete()*. Kode Sumber untuk menghapus data ditunjukkan pada kode sumber B.6.

4.5.6 Melihat Detail Submodul Modul General Ledger

Pada implementasi melihat detail *Chart of Accounts*, *Journals,* dan *Period,* sistem menampilkan detail akun yang disimpan dalam basis data. Pengaturan melihat detail customer diatur pada fungsi `actionView()` yang ditunjukkan pada Kode Sumber B.3.

4.5.7 Melihat Tipe Laporan

Pada implementasi melihat laporan *Balance Sheet, Income Statement,* dan *Cash Flow,* sistem menampilkan modal yang berisi tipe-tipe laporan yang dapat ditampilkan. Pengaturan tampilan ini diatur pada fungsi `actionIndexajax()`. Kode untuk menampilkan modal tersebut ditunjukkan pada kode sumber B.7.

4.5.8 Melihat Laporan *Balance Sheet*

Pada implementasi melihat laporan *balance sheet* secara tahunan, sistem akan menampilkan daftar akun beserta debit, kreditnya. Pengaturan tampilan ini diatur pada kelas *BsheetController* fungsi *actionYearly().* Kode untuk menampilkan seluruh data tersebut ditunjukkan pada kode sumber B.8.

4.5.9 Melihat Laporan *Income Statement*

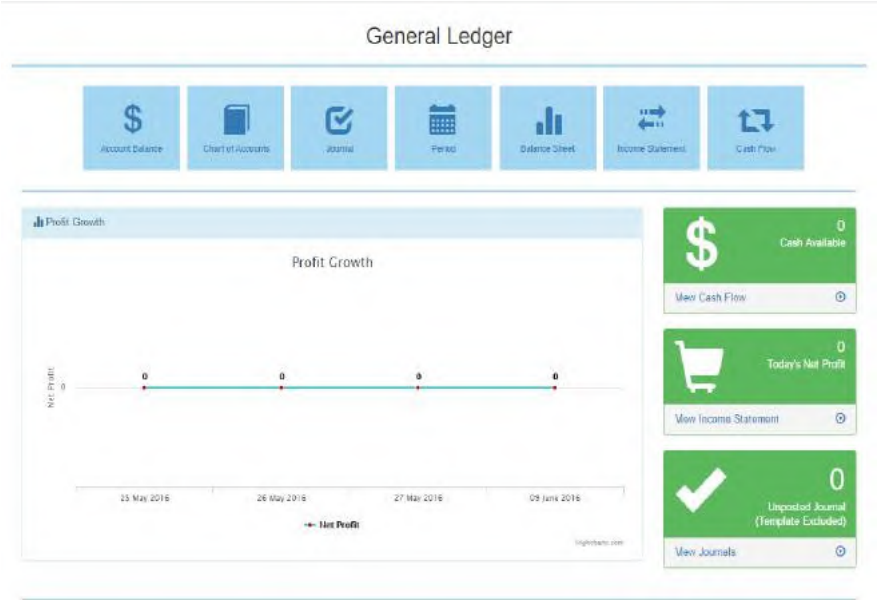
Pada implementasi melihat laporan *income statement* secara tahunan, sistem akan menampilkan daftar akun beserta debit, kreditnya. Pengaturan tampilan ini diatur pada kelas *IstatementController* fungsi *actionYearly().* Kode untuk menampilkan seluruh data tersebut ditunjukkan pada kode sumber B.9.

4.5.10 Melihat Laporan *Cash Flow*

Pada implementasi melihat laporan *cash flow* tahunan, sistem akan menampilkan daftar akun beserta debit, kreditnya. Pengaturan tampilan ini diatur pada kelas *CflowController* fungsi *actionYearly().* Kode untuk menampilkan seluruh data tersebut ditunjukkan pada kode sumber B.10.

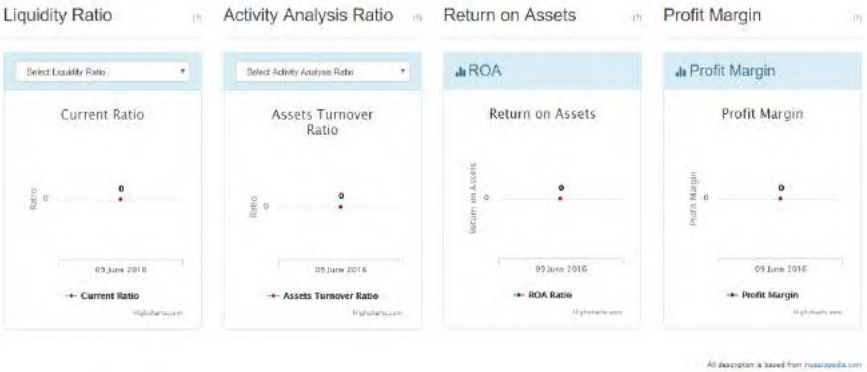
4.5.11 Antarmuka Halaman Utama Modul *General Ledger*

Pada antarmuka ini pengguna dapat melihat grafik perkembangan keuntungan. Pengguna juga dapat melihat kas yang ada sekarang serta keuntungan yang telah didapat hari ini. Pengguna juga diberikan analisis dari laporan keuangan yang ada. Tampilan antarmuka ini dapat dilihat pada Gambar 4.1. dan Gambar 4.2.



Gambar 4.1. Antarmuka Halaman Utama Modul *General Ledger* (1)

Pada Gambar 4.1. digambarkan keadaan keuangan saat ini serta ditampilkan beberapa sub-modul yang ada dari modul *accounting*. Grafik ditengah berguna untuk menampilkan perkembangan keuntungan yang diraih oleh perusahaan. Terdapat 3 kotak di sebelah kanan, kotak yang pertama menampilkan uang yang tersedia saat ini, kotak kedua menampilkan keuntungan yang sudah diraih pada hari ini, sedangkan kotak ketiga menampilkan jurnal-jurnal yang belum diverifikasi atau di-*post* ke dalam buku besar.

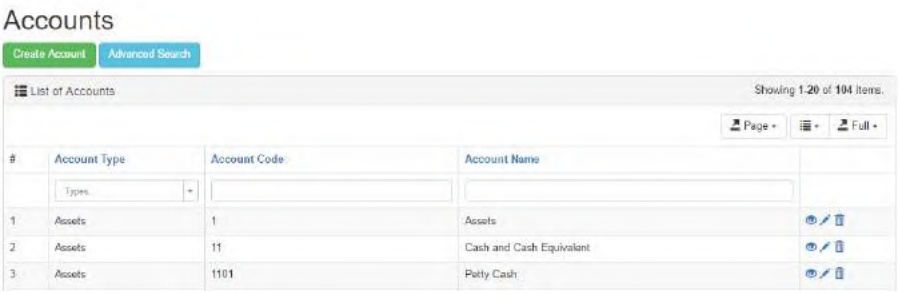


Gambar 4.2. Antarmuka Halaman Utama Modul *General Ledger* (2)

Pada Gambar 4.2. ditampilkan beberapa analisis dari laporan keuangan berupa rasio-rasio. Terdapat *liquidity ratio, activity analysis ratio, return on assets*, dan *profit margin.*

4.5.12 Antarmuka Melihat Data Chart of Accounts

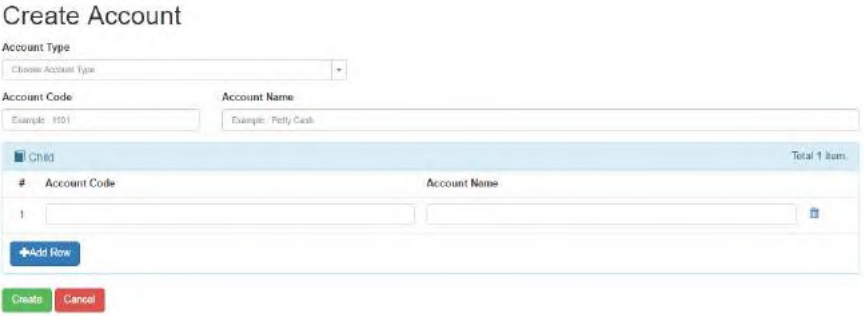
Pada antarmuka ini pengguna dapat melihat daftar *Chart of Accounts* yang terdapat pada sistem. Pengguna juga dapat memilih untuk menyunting, melihat rincian, atau menghapus *Account* tertentu serta membuat *Account* baru. Tampilan antarmuka ini dapat dilihat pada Gambar 4.3.



Gambar 4.3. Antarmuka Melihat Data *Chart of Accounts*

Gambar 4.3. yang berisi tentang antarmuka dari kasus penggunaan "Melihat Data *Chart of Accounts*" berisikan tabel yang menampilkan data-data yang ada di *database* dari *chart of account.* Terdapat 3 kolom utama pada tabel tersebut. *Account Type* berisikan tipe dari akun tersebut, pada sistem sudah terdapat 5 tipe akun untuk membedakan masing-masing akun yaitu: *assets, liabilites, equities, revenues,* dan *expenses. Account Code* berisikan kode dari masing-masing akun, seperti pada baris ke-dua pada Gambar 4.3. terdapat *account code* "11" untuk akun yang bernama *Cash and Cash Equivalent*. *Account Name* berisikan nama dari masing-masing akun.

4.5.13 Antarmuka Menambah Data Chart of Accounts



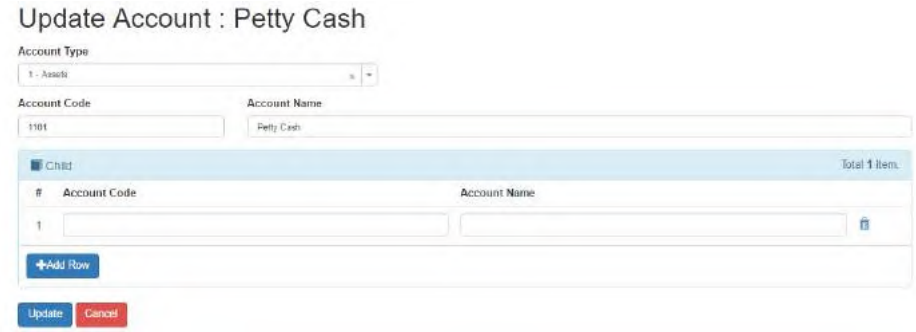
Gambar 4.4. Antarmuka Menambah Data *Chart of Accounts*

Pada antarmuka ini pengguna dapat mengisi data yang dibutuhkan terkait *Account* dan menekan tombol "*Create*" untuk perintah pembuatan. Tampilan antarmuka yang berupa form ini dapat dilihat pada Gambar 4.4.

Gambar 4.4. merupakan antarmuka dari kasus penggunaan "Menambah Data *Chart of Accounts*". Terdapat 4 aspek utama dari antarmuka ini yaitu *account type, account code, account name,* serta tabel *child*. *Account Type* berisikan tipe dari akun tersebut, pada sistem sudah terdapat 5 tipe akun untuk membedakan masing-masing akun yaitu: *assets, liabilites, equities, revenues,* dan *expenses. Account Code* adalah kode dari akun yang akan dibuat, kode dibuat berdasarkan kesepakatan bersama dari perusahaan sehingga data bisa dibuat sesuai dengan perusahaan masing-masing, begitu juga dengan *Account Name.*

Tabel *Child* berisi data-data yang sama dengan data *parent* yang diisi sebelumnya, namun tidak perlu mengisi *account type* karena sesuai dengan *parent* tersebut. Tabel *Child* berisi akun turunan (pecahan) dari yang telah dibuat sebagai *parent*. Kode akun dari *child* tersebut juga merupakan kesepakatan perusahaan pada awalnya, beberapa perusahaan menggunakan kode akun *child* yang hampir sama dengan *parent*-nya, sebagai contoh apabila parent memiliki kode akun "11" maka *child*-nya memiliki kode akun seperti "111", "112" dan seterusnya.

4.5.14 Antarmuka Menyunting Data Chart of Accounts



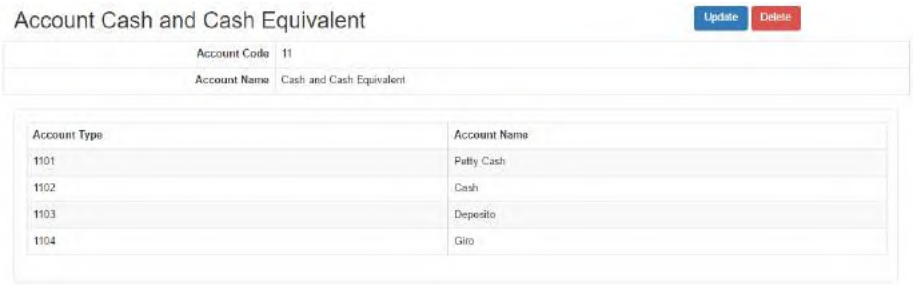
Gambar 4.5. Antarmuka Menyunting Data *Chart of Accounts*

Pada antarmuka yang ditampilkan pada Gambar 4.5 ini pengguna dapat merubah data *Account* dan menekan tombol "*Update*" untuk perintah penyimpanan perubahan data. Terdapat 4 aspek utama dari antarmuka ini yaitu *account type*, *account code*, *account name*, serta tabel *child*. *Account Type* berisikan tipe dari akun tersebut, pada sistem sudah terdapat 5 tipe akun untuk membedakan masing-masing akun yaitu: *assets*, *liabilites*, *equities*, *revenues*, dan *expenses. Account Code* adalah kode dari akun yang akan dibuat, kode dibuat berdasarkan kesepakatan bersama dari perusahaan sehingga data bisa dibuat sesuai dengan perusahaan masing-masing, begitu juga dengan *Account Name.*

Tabel *Child* berisi data-data yang sama dengan data *parent* yang diisi sebelumnya, namun tidak perlu mengisi *account type* karena sesuai dengan *parent* tersebut. Tabel *Child* berisi akun turunan (pecahan) dari yang telah dibuat sebagai *parent*. Kode akun dari *child* tersebut juga merupakan kesepakatan perusahaan pada awalnya, beberapa perusahaan menggunakan kode akun *child* yang hampir sama dengan *parent*-nya, sebagai contoh apabila parent memiliki kode akun "11" maka *child*-nya memiliki kode akun seperti "111", "112" dan seterusnya.

4.5.15 Antarmuka Melihat Detail Chart of Accounts

Pada antarmuka ini pengguna dapat melihat detail dari *chart of account* yang dipilih. Tampilan antarmuka ini dapat dilihat pada Gambar 4.6.

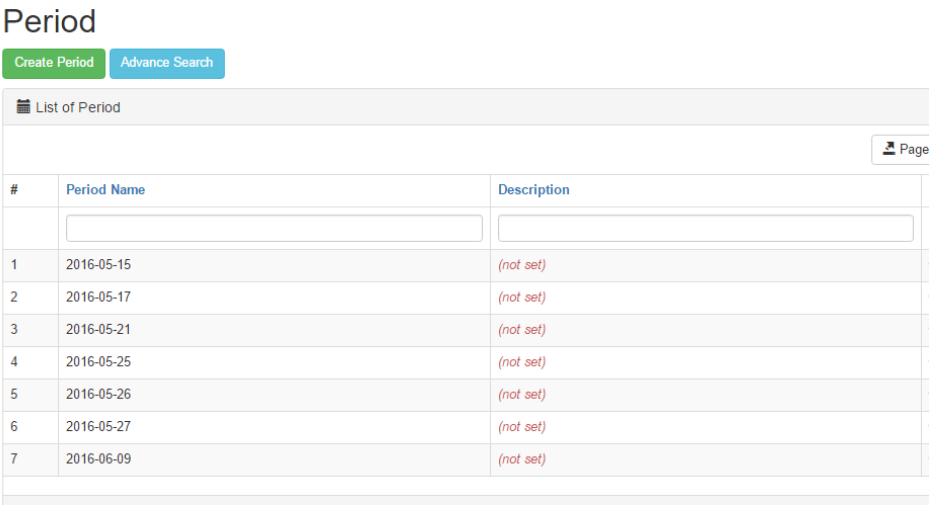


Gambar 4.6. Antarmuka Melihat Detail *Chart of Accounts*

Gambar 4.6. yang merupakan implementasi antarmuka dari kasus penggunaan "Melihat Detail *Chart of Accounts*". Pada antarmuka ini ditampilkan anak-anak dari akun tersebut. Sebagai contoh pada Gambar 4.6, akun *cash and cash equivalent* memiliki 4 anak yaitu *petty cash* ,*cash*, deposito, dan giro.

4.5.16 Antarmuka Melihat Data *Period*

Pada antarmuka ini pengguna dapat melihat daftar *Period* yang terdapat pada sistem. Pengguna juga dapat memilih untuk menyunting, melihat rincian, atau menghapus *Period* tertentu serta membuat *Period* baru. Tampilan antarmuka ini dapat dilihat pada Gambar 4.7.



Gambar 4.7. Antarmuka Melihat Data *Period*

4.5.17 Antarmuka Menambah Data *Period*

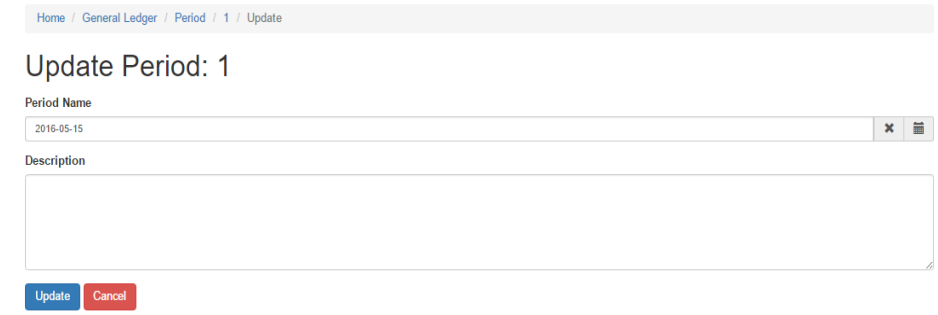
Pada antarmuka ini pengguna dapat mengisi data yang dibutuhkan terkait *Period* dan menekan tombol "*Create*" untuk perintah pembuatan. Tampilan antarmuka yang berupa form ini dapat dilihat pada Gambar 4.8.

Gambar 4.8. Antarmuka Menambah Data *Period*

Gambar 4.8. yang merupakan implementasi antarmuka dari kasus penggunaan "Menambah Data *Period*", memiliki 2 form utama yaitu *Period Name* dan *Description*. *Period Name* diisi dengan tanggal yang akan dibuat periodenya, sedangkan *description* berisikan deskripsi dari periode tersebut.

4.5.18 Antarmuka Menyunting Data *Period*

Pada antarmuka ini pengguna dapat merubah data *Period* dan menekan tombol "*Update*" untuk perintah penyimpanan perubahan data. Tampilan antarmuka ini dapat dilihat pada lampiran Gambar 4.9.

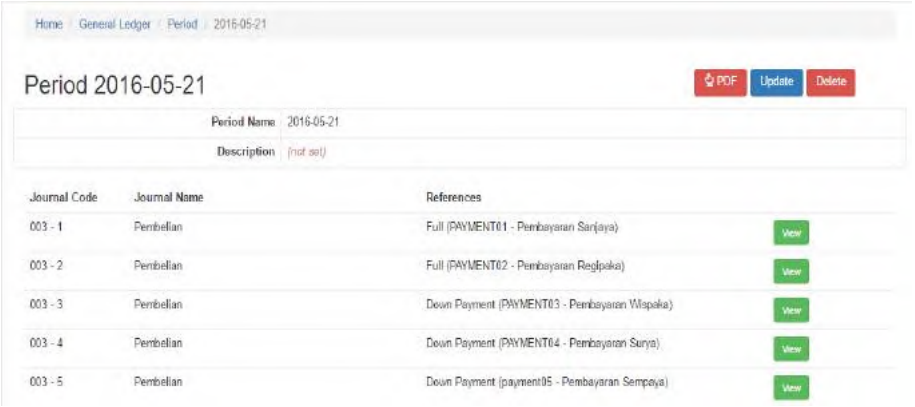


Gambar 4.9. Antarmuka Menyunting Data *Period*

Gambar 4.9. yang merupakan implementasi antarmuka dari kasus penggunaan "Menyunting Data *Period*", memiliki 2 form utama yaitu *Period Name* dan *Description*. *Period Name* diisi dengan tanggal yang akan dibuat periodenya, sedangkan *description* berisikan deskripsi dari periode tersebut.

4.5.19 Antarmuka Melihat Detail *Period*

Pada antarmuka ini pengguna dapat melihat detail dari *period* yang dipilih. Tampilan antarmuka ini dapat dilihat pada Gambar 4.10.



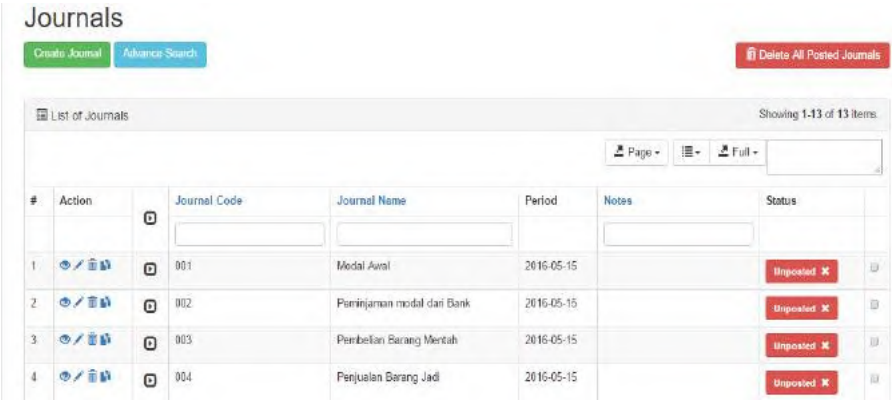
Gambar 4.10. Antarmuka Melihat Detail *Period*

Gambar 4.10. yang merupakan implementasi antarmuka dari kasus penggunaan "Melihat Detail *Period*". Disini tertera jurnal-jurnal yang berhubungan dengan periode tersebut. Seperti contoh pada Gambar 4.10. terdapat 5 jurnal yang dibuat pada periode 2016-05-21.

4.5.20 Antarmuka Melihat Data *Journal*

Pada antarmuka ini pengguna dapat melihat daftar *Journal* yang terdapat pada sistem. Pengguna juga dapat memilih untuk menyunting, melihat rincian, atau menghapus *Account* tertentu serta membuat

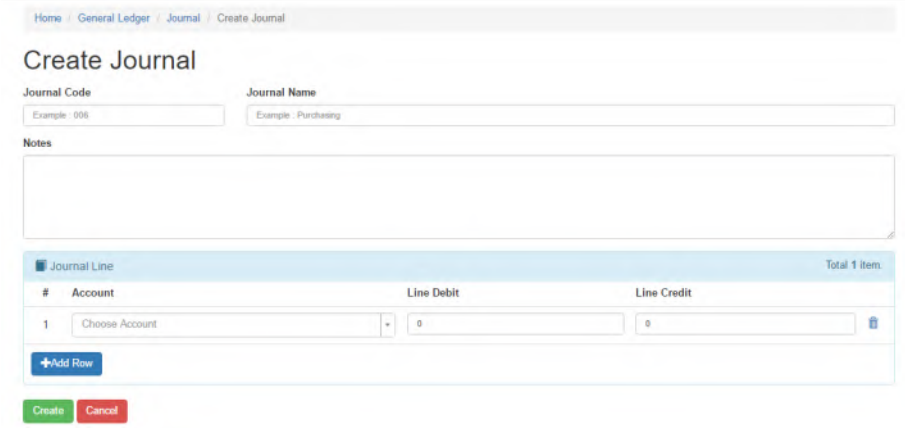
Account baru. Tampilan antarmuka ini dapat dilihat pada Gambar 4.11.



Gambar 4.11. Antarmuka Melihat Data *Journals*

Gambar 4.11. yang merupakan implementasi antarmuka dari kasus penggunaan "Melihat Data *Journals"*. Tampilan ini berisikan data-data jurnal yang ada di *database journal* dan ditampilkan dalam bentuk tabel. Terdapat 7 kolom pada tampilan ini yaitu *action, expand, journal code, journal name, period, notes,* dan *status. Action* berisikan *action* apa saja yang bisa dilakukan dengan jurnal tersebut, kolom *action* ini sebenarnya sama dengan kasus penggunaan lainnya, namun di jurnal terdapat satu *action* tambahan yaitu *duplicate* dan hanya bisa dilakukan untuk jurnal yang *unposted*. *Expand* yang merupakan tanda kotak berisikan panah, menampilkan data dari jurnal tersebut. *Journal Code* merupakan kode jurnal yang digunakan oleh perusahaan (kode merupakan kebijakan perusahaan masing-masing, begitu juga dengan *journal name*. *Period* berisikan tentang tanggal berapa jurnal tersebut dibuat. *Notes* adalah tambahan apabila ada catatan tambahan yang dapat dimasukkan ke dalam jurnal tersebut. *Status* berisikan 2 data, *Posted* dan *Unposted*, apabila jurnal tersebut *posted*, maka jurnal tersebut akan dimasukkan ke dalam laporan keuangan, namun tidak untuk *unposted*. Tombol *Posted* dan *Unposted* dapat ditekan apabila ingin merubah status dari jurnal tersebut.

4.5.21 Antarmuka Menambah Data *Journal*

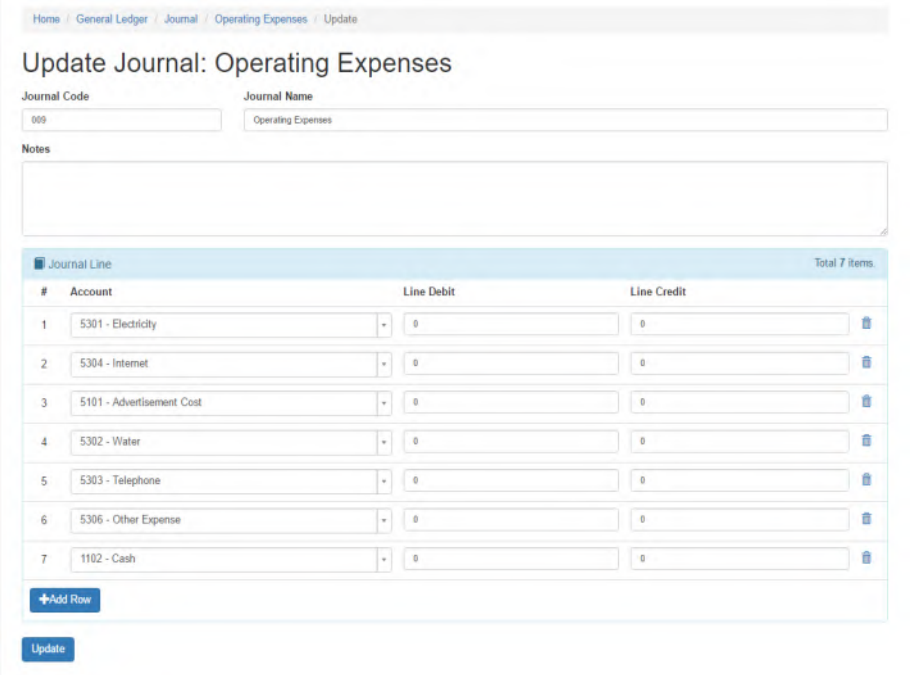


Gambar 4.12. Antarmuka Menambah Data *Journal*

Pada antarmuka yang ditampilkan pada Gambar 4.12 ini pengguna dapat mengisi data yang dibutuhkan terkait *Journal* dan menekan tombol "*Create*" untuk perintah pembuatan. Ada juga form *Journal Code* dan *Journal Name* yang dapat diisikan sesuai dengan kebijakan perusahaan. *Journal Line* berisikan akun mana saja yang berhubungan dengan jurnal yang dibuat, pada *journal line* juga terdapat *line debit* dan *line credit*, kedua kolom tersebut digunakan apabila ada terjadi transaksi dan dimasukkan dalam bentuk angka. Total dari *Line Debit* dan *Line Credit* harus seimbang demi menyeimbangkan *Balance Sheet* nantinya.

4.5.22 Antarmuka Menyunting Data *Journal*

Pada antarmuka ini pengguna dapat merubah data *Journal* dan menekan tombol "*Update*" untuk perintah penyimpanan perubahan data. Tampilan antarmuka ini dapat dilihat pada lampiran Gambar 4.13.



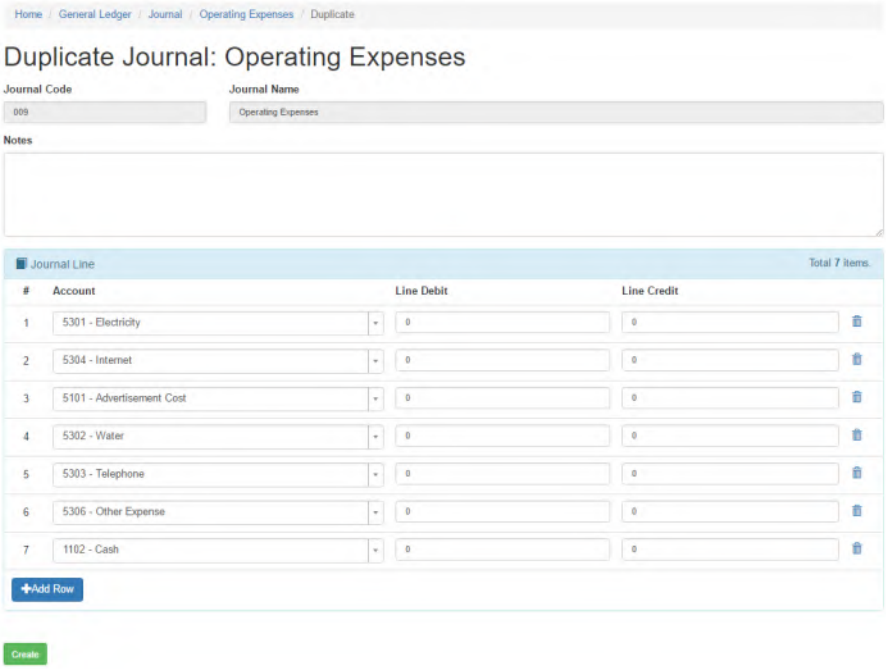
Gambar 4.13. Antarmuka Menyunting Data *Journal*

Pada Gambar 4.13. ada juga form *Journal Code* dan *Journal Name* yang dapat diisikan sesuai dengan kebijakan perusahaan. *Journal Line* berisikan akun mana saja yang berhubungan dengan jurnal yang dibuat, pada *journal line* juga terdapat *line debit* dan *line credit*, kedua kolom tersebut digunakan apabila ada terjadi transaksi dan dimasukkan dalam bentuk angka. Total dari *Line Debit* dan *Line Credit* harus seimbang demi menyeimbangkan *Balance Sheet* nantinya.

4.5.23 Antarmuka Menduplikasi Data *Journal*

Pada antarmuka ini pengguna dapat menduplikasi data dari header yang dibutuhkan terkait *Journal* dan menekan tombol

"*Duplicate*" untuk perintah penduplikasian. Tampilan antarmuka yang berupa form ini dapat dilihat pada Gambar 4.14.

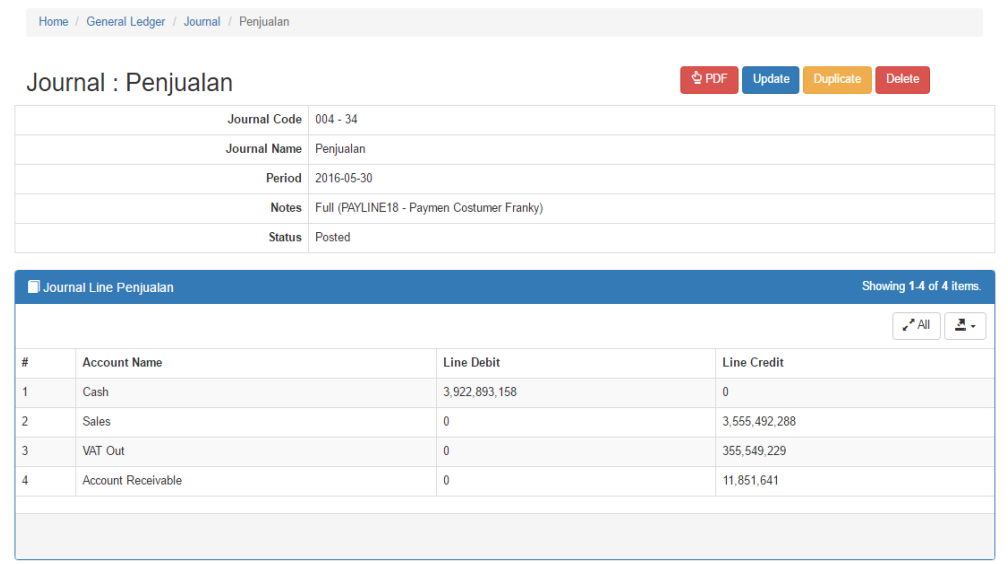


Gambar 4.14. Antarmuka Menduplikasi Data *Journal*

Pada Gambar 4.14. ada juga form *Journal Code* dan *Journal Name*, namun pada duplikasi, kedua form tersebut tidak bisa diisi. *Journal Line* berisikan akun mana saja yang berhubungan dengan jurnal yang dibuat, pada *journal line* juga terdapat *line debit* dan *line credit*, kedua kolom tersebut digunakan apabila ada terjadi transaksi dan dimasukkan dalam bentuk angka. Total dari *Line Debit* dan *Line Credit* harus seimbang demi menyeimbangkan *Balance Sheet* nantinya.

4.5.24 Antarmuka Melihat Detail *Journal*

Pada antarmuka ini pengguna dapat melihat detail dari *journal* yang dipilih. Tampilan antarmuka ini dapat dilihat pada Gambar 4.15.

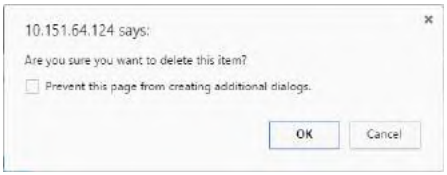


Gambar 4.15. Antarmuka Melihat Detail *Journal*

Gambar 4.15. yang merupakan implementasi antarmuka dari kasus penggunaan "Melihat Detail *Journal*", menampilkan data yang ada dari satu jurnal yang dipilih idnya. Semisal pada Gambar 4.15. pada jurnal "Penjualan" terdapat semua data terkait dengan jurnal tersebut. Pada kasus ini, jurnal "penjualan" memang juga ada yang lain, tetapi dapat dilihat di *journal code* terdapat " – 34" yang berarti jurnal penjualan yang dipilih merupakan jurnal penjualan yang ke 34. Pada tabel dibawah juga dijabarkan terkait line apa saja yang ada pada jurnal tersebut dan bagaimana debit dan kreditnya dari masing-masing akun.

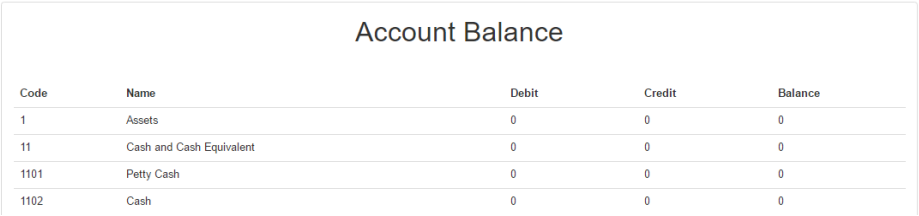
4.5.25 Antarmuka Menghapus Data Submodul

Pada antarmuka ini ditampilkan halaman berupa *pop-up* untuk melakukan konfirmasi penghapusan data dengan menekan tombol "*Yes*" atau pembatalan dengan menekan tombol "*Cancel*". Tampilan antarmuka ini dapat dilihat pada lampiran Gambar 4.16.



Gambar 4.16. Antarmuka Menghapus Data Submodul

4.5.26 Antarmuka Melihat Data *Account Balance*



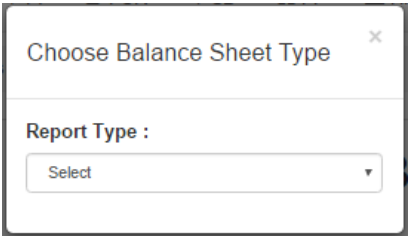
Gambar 4.17. Antarmuka Melihat Data *Account Balance*

Gambar 4.17. merupakan implementasi antarmuka dari kasus penggunaan "Melihat Data *Account Balance*". Pada antarmuka ini ditampilkan detil dari masing-masing akun, berapa debit serta kredit yang telah dilakukan dari awal periode sampai sekarang ini semua terekam, sehingga dapat membantu akuntan dalam menelusuri kesalahan pada laporan keuangan.

4.5.27 Antarmuka Melihat Tipe Laporan *Balance Sheet*

Pada antarmuka ini pengguna dapat melihat modal yang berisi tipe-tipe *Balance Sheet* yang terdapat pada sistem (*Annual, Monthly*,

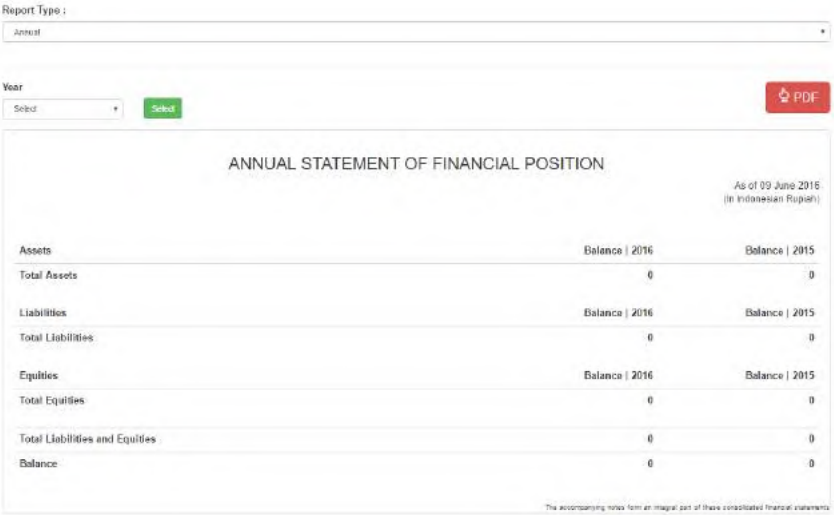
Daily, *Quarterly*). Tampilan antarmuka ini dapat dilihat pada Gambar 4.18.



Gambar 4.18. Antarmuka Melihat Tipe Laporan *Balance Sheet*

4.5.27.1 Antarmuka Melihat Laporan *Balance Sheet* Tahunan

Pada antarmuka ini pengguna dapat melihat laporan *balance sheet* yang bertipe tahunan berdasarkan jurnal yang terdapat pada sistem. Tampilan antarmuka ini dapat dilihat pada Gambar 4.19.

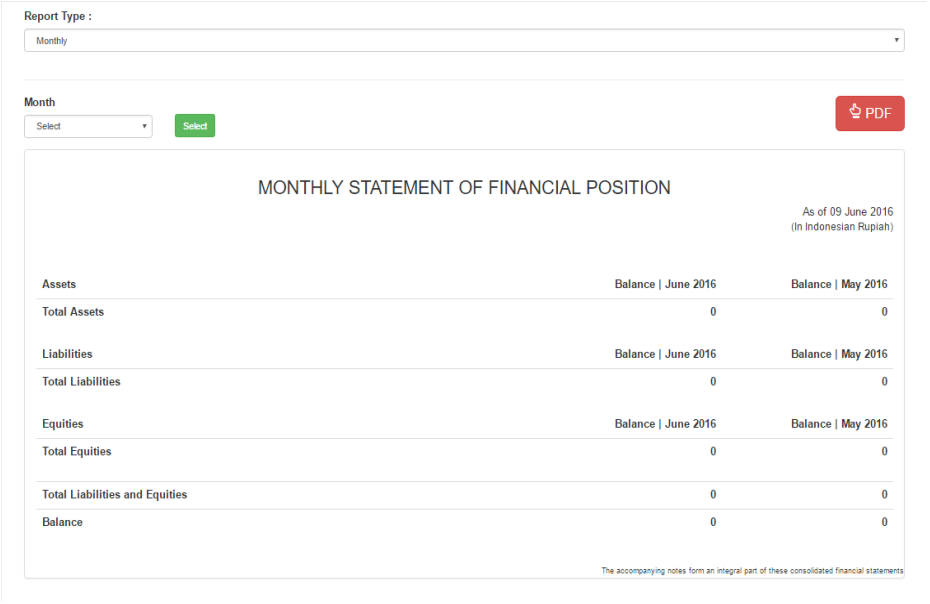


Gambar 4.19. Antarmuka Melihat Laporan *Balance Sheet* Tahunan

Balance Sheet memiliki 3 aspek utama yaitu *assets*, *liabilities*, dan *equities*. *Balance Sheet* dapat dikatakan *"balance"* apabila jumlah *assets* sama dengan jumlah *liabilities* ditambahkan dengan *equities*.

4.5.27.2 Antarmuka Melihat Laporan *Balance Sheet* Bulanan

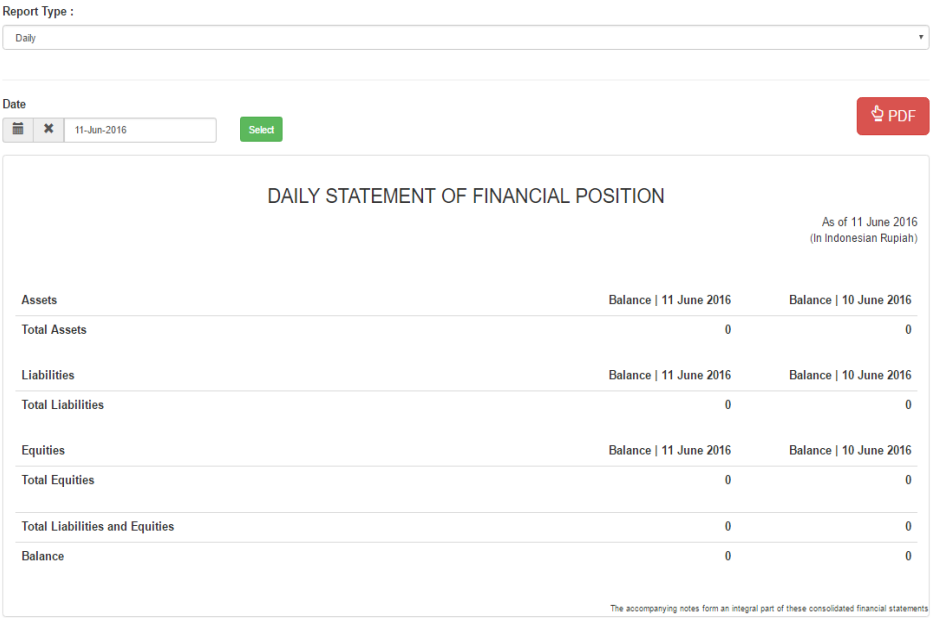
Pada antarmuka ini pengguna dapat melihat laporan *balance sheet* yang bertipe bulanan berdasarkan jurnal yang terdapat pada sistem. Tampilan antarmuka ini dapat dilihat pada Gambar 4.20.



Gambar 4.20. Antarmuka Melihat Laporan *Balance Sheet* Bulanan

4.5.27.3 Antarmuka Melihat Laporan *Balance Sheet* Harian

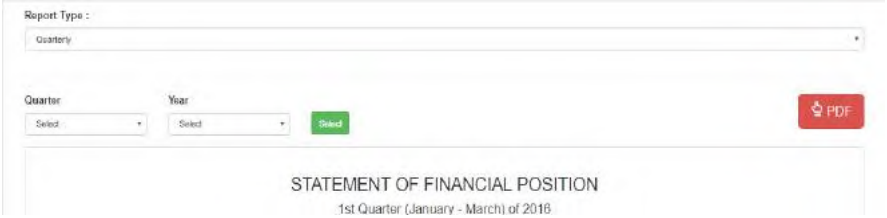
Pada antarmuka ini pengguna dapat melihat laporan *balance sheet* yang bertipe harian berdasarkan jurnal yang terdapat pada sistem. Tampilan antarmuka ini dapat dilihat pada Gambar 4.21.

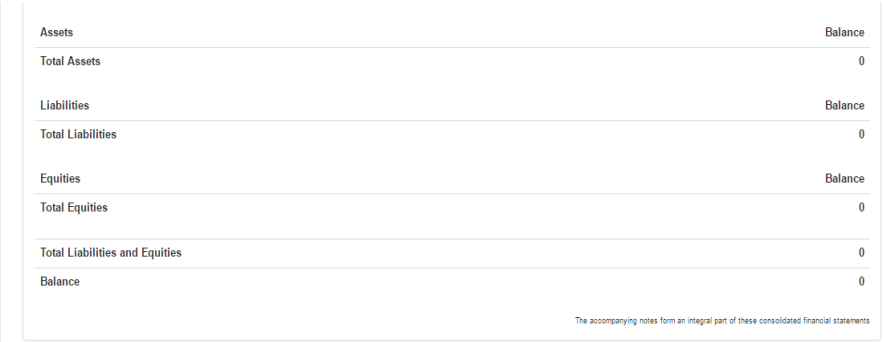


Gambar 4.21. Antarmuka Melihat Laporan *Balance Sheet* Harian

4.5.27.4 Antarmuka Melihat Laporan *Balance Sheet* Periode Tiga Bulanan

Pada antarmuka ini pengguna dapat melihat laporan *balance sheet* yang bertipe periode 3 (tiga) bulanan berdasarkan jurnal yang terdapat pada sistem. Tampilan antarmuka ini dapat dilihat pada Gambar 4.22.





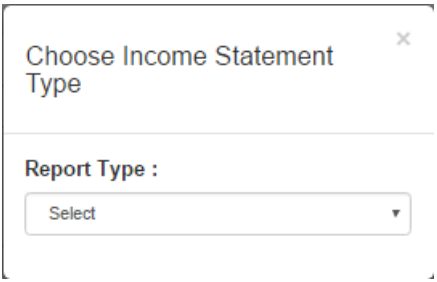
Assets	Balance
Total Assets	0
Liabilities	**Balance**
Total Liabilities	0
Equities	**Balance**
Total Equities	0
Total Liabilities and Equities	0
Balance	0

The accompanying notes form an integral part of these consolidated financial statements

Gambar 4.22. Antarmuka Melihat Laporan *Balance Sheet* Tiga Bulanan

4.5.28 Antarmuka Melihat Tipe Laporan *Income Statement*

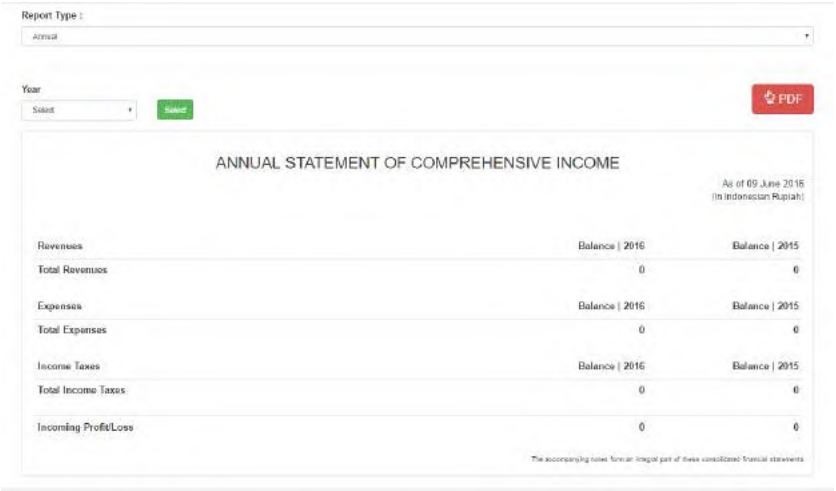
Pada antarmuka ini pengguna dapat melihat modal yang berisi tipe-tipe *income statement* yang terdapat pada sistem. Tampilan antarmuka ini dapat dilihat pada Gambar 4.23.



Gambar 4.23. Antarmuka Melihat Tipe Laporan *Income Statement*

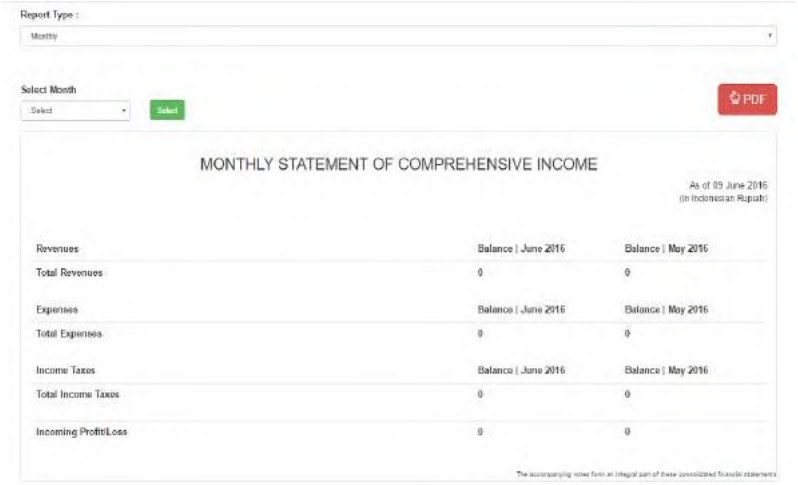
4.5.28.1 Antarmuka Melihat Laporan *Income Statement* Tahunan

Pada antarmuka ini pengguna dapat melihat laporan *income statement* yang bertipe tahunan berdasarkan jurnal yang terdapat pada sistem. Tampilan antarmuka ini dapat dilihat pada Gambar 4.24.



Gambar 4.24. Antarmuka Melihat Laporan *Income Statement* Tahunan

4.5.28.2 Antarmuka Melihat Laporan *Income Statement* Bulanan

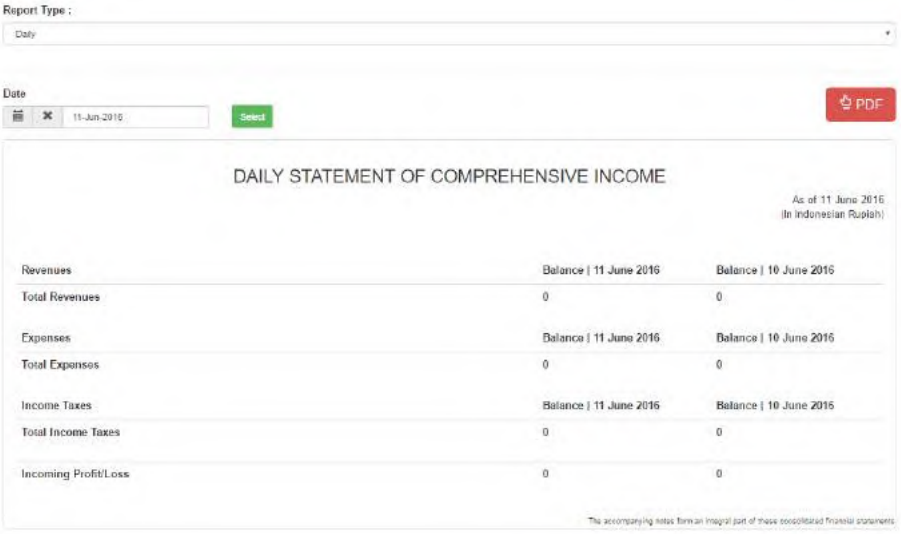


Gambar 4.25. Antarmuka Melihat Laporan *Income Statement* Bulanan

Pada antarmuka ini pengguna dapat melihat laporan *income statement* yang bertipe bulanan berdasarkan jurnal yang terdapat pada sistem. Tampilan antarmuka ini dapat dilihat pada Gambar 4.25.

4.5.28.3 Antarmuka Melihat Laporan *Income Statement* Harian

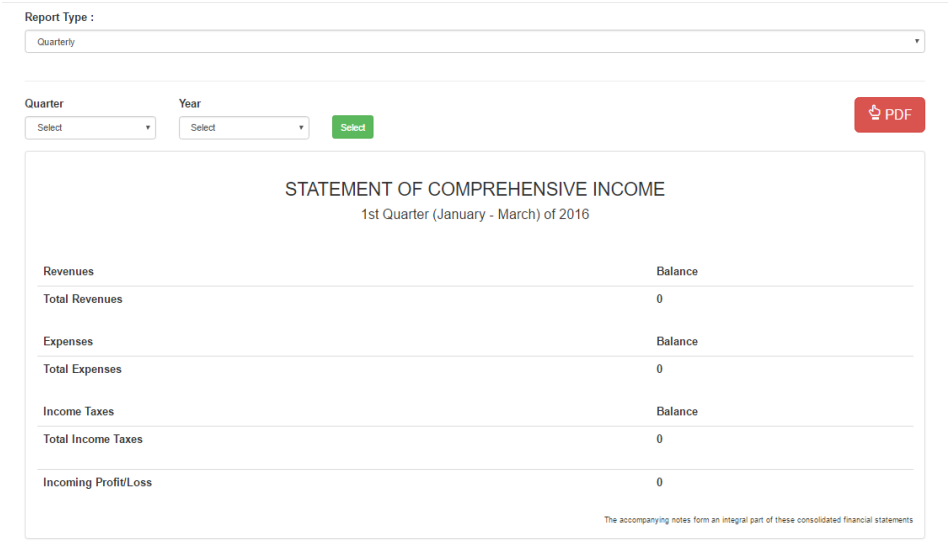
Pada antarmuka ini pengguna dapat melihat laporan *balance sheet* yang bertipe harian berdasarkan jurnal yang terdapat pada sistem. Tampilan antarmuka ini dapat dilihat pada Gambar 4.26.



Gambar 4.26. Antarmuka Melihat Laporan *Income Statement* Harian

4.5.28.4 Antarmuka Melihat Laporan *Income Statement* Periode Tiga Bulanan

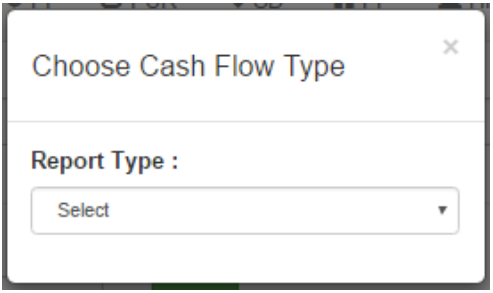
Pada antarmuka ini pengguna dapat melihat laporan *balance sheet* yang bertipe periode 3 (tiga) bulanan berdasarkan jurnal yang terdapat pada sistem. Tampilan antarmuka ini dapat dilihat pada Gambar 4.27.



Gambar 4.27. Antarmuka Melihat Laporan *Income Statement* 3 Bulanan

4.5.29 Antarmuka Melihat Tipe Laporan *Cash Flow*

Pada antarmuka ini pengguna dapat melihat modal yang berisi tipe-tipe *cash flow* yang terdapat pada sistem. Tampilan antarmuka ini dapat dilihat pada Gambar 4.28.



Gambar 4.28. Antarmuka Melihat Tipe Laporan *Cash Flow*

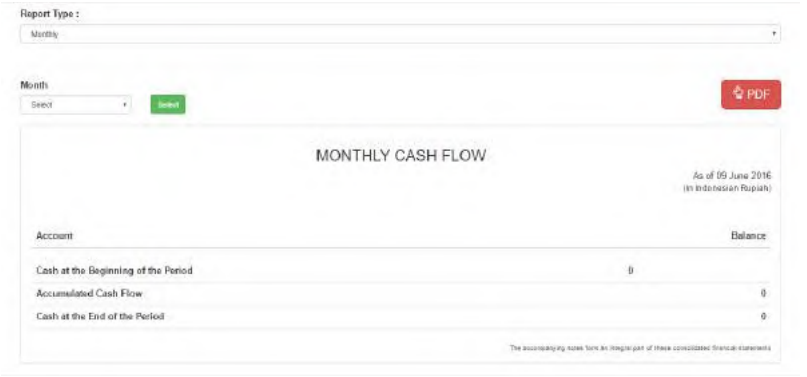
4.5.29.1 Antarmuka Melihat Laporan *Cash Flow* Tahunan

Pada antarmuka ini pengguna dapat melihat laporan *cash flow* yang bertipe tahunan berdasarkan jurnal yang terdapat pada sistem. Tampilan antarmuka ini dapat dilihat pada Gambar 4.29.

Gambar 4.29. Antarmuka Melihat Laporan *Cash Flow* Tahunan

4.5.29.2 Antarmuka Melihat Laporan *Cash Flow* Bulanan

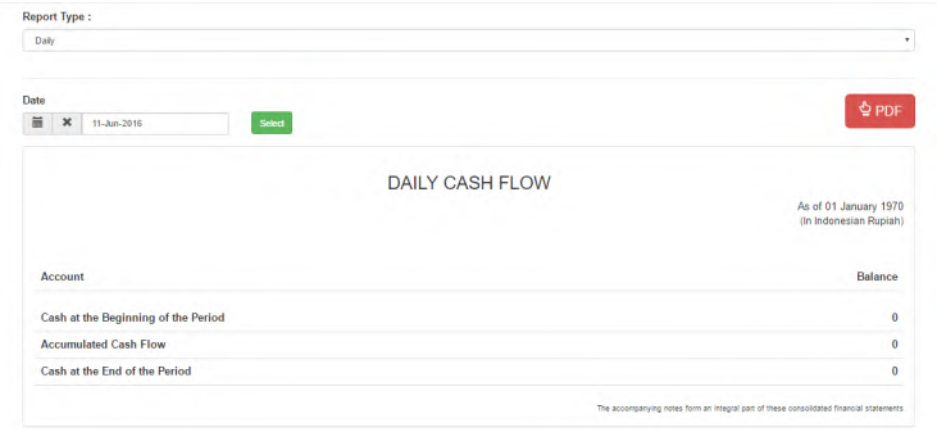
Pada antarmuka ini pengguna dapat melihat laporan *cash flow* yang bertipe bulanan berdasarkan jurnal yang terdapat pada sistem. Tampilan antarmuka ini dapat dilihat pada Gambar 4.30.



Gambar 4.30. Antarmuka Melihat Laporan *Cash Flow* Bulanan

4.5.29.3 Antarmuka Melihat Laporan *Cash Flow* Harian

Pada antarmuka ini pengguna dapat melihat laporan *balance sheet* yang bertipe harian berdasarkan jurnal yang terdapat pada sistem. Tampilan antarmuka ini dapat dilihat pada Gambar 4.31.



Gambar 4.31. Antarmuka Melihat Laporan *Cash Flow* Harian

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas hasil dan pembahasan pada aplikasi yang dikembangkan. Pada bab ini akan dijelaskan tentang data yang digunakan, hasil yang didapatkan dari penggunaan perangkat lunak dan uji coba yang dilakukan pada perangkat lunak yang telah dikerjakan untuk menguji apakah fungsionalitas aplikasi telah diimplementasikan dengan benar dan berjalan sebagaimana mestinya.

5.1 Lingkungan Pengujian

Lingkungan uji coba menjelaskan lingkungan yang digunakan untuk menguji implementasi pembuatan sistem pada tugas akhir ini. Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang dijelaskan sebagai berikut:

1. Database yang digunakan pada server adalah MySQL Cluster.
2. 2 PC untuk server database menggunakan Ubuntu 12.04.
3. PC untuk server menggunakan Intel® Core™ i3-2120 @3.30GHz , RAM 4GB dengan Sistem Operasi Windows 8.1 Enterprise x64.
4. Mozilla Firefox 46.0.1 dan Chrome 49.0 sebagai antarmuka untuk pengujian aplikasi klien.

5.2 Skenario Pengujian

Pada bagian ini akan dijelaskan tentang skenario pengujian yang dilakukan. Pengujian dilakukan dalam tiga tahap yaitu pengujian kebutuhan fungsionalitas, pengujian kegunaan sistem, dan pengujian perbandingan *metrics*. Pengujian kebutuhan fungsionalitas menggunakan metode kotak hitam (*black box*). Metode ini menekankan pada hasil keluaran sistem.

Prosedur Simulasi Pasar

- **Peserta**

5 perusahaan (45 orang bisa lebih) yang terdiri dari Perusahaan A, Perusahaan B, Perusahaan C, Perusahaan D dan Perusahaan E.

- **Aturan Simulasi**
  - **Make to Stock**
    1. Perusahaan menentukan sendiri supplier dari bahan baku yang akan dibeli berdasarkan data per supplier (histori defect rate, harga barang, toleransi pembayaran, denda dan diskon) yang diberikan pasar.
    2. Perusahaan melakukan produksi berdasarkan hasil forecasting dari histori serapan.
    3. Perusahaan menentukan sendiri strategi penyebaran finished goods nya pada pasar yang tersedia (Grosir, Midi, Retail).
    4. Perusahaan menyiapkan COGS dan biaya iklan yang diberikan kepada pasar yang akan digunakan pasar untuk menentukan serapan.
    5. Perusahaan menjual finished goods nya sesuai serapan yang diberikan oleh pasar.

  - **Make to Order**
    1. Perusahaan menentukan sendiri supplier dari bahan baku yang akan dibeli berdasarkan data per supplier (histori defect rate, harga barang, toleransi pembayaran, denda dan diskon) yang diberikan pasar.
    2. Perusahaan menyiapkan COGS dan biaya iklan yang diberikan kepada pasar yang akan digunakan pasar untuk menentukan serapan.
    3. Perusahaan melakukan produksi (apabila diperlukan) hingga finished goods sesuai dengan serapan yang diberikan oleh pasar.
    4. Perusahaan menjual finished goods nya sesuai serapan yang diberikan oleh pasar.

- ***Business Plan***
  - ***Finished Goods***

Tabel 5.1. *Finished Goods* Data

No	Nama Barang

1	Black Deluxe Touring Bike
2	Black Profesional Touring Bike

Perusahaan ini memproduksi dua jenis sepeda. Dua jenis sepeda tersebut memiliki bahan dasar yang berbeda, *Deluxe Touring Bike* berbahan dasar aluminum, sedangkan *Professional Touring Bike* berbahan dasar karbon. Keduanya memiliki warna hitam.

- ***Assets***

Tabel 5.2 *Assets* Data

No.	Name	Quantity
1	Tanah	180x150 m2
2	Kantor	50x50 m2
3	Parkir	60x50 m2
4	Kantin	40x60 m2
5	Pabrik	108x80 m2
6	Pengolahan Limbah	20x40 m2
7	*Raw Materials Inventory*	40x60 m2
8	*Semi-Finished Good Inventory*	40x60 m2
9	*Finished Good Inventory*	40x60 m2
10	*Welding Machine*	2 line
11	*Laser Cutting Machine*	2 line
12	*Testing Machine*	2 line
13	*Truck*	10
14	*Forklift*	6
15	*Heavy Forklift*	6

Perusahaan ini memiliki 15 asset pada perencanaannya. Dijabarkan pada tabel 5.2. berikut dengan kuantitas dari masing-masing aset yang ada.

- ***Raw Materials***

Tabel 5.3. *Raw Materials* Data

No.	Raw Materials Name	Quantity	UoM
1	Tire	1.2	Pcs

No.	Raw Materials Name	Quantity	UoM
2	Seat Kit	0.5	Pcs
3	Chain	1.5	Pcs
4	Gear	1.8	Pcs
5	Brake	0.6	Pcs
6	Handle Bar	1.2	Pcs
7	Pedal	0.4	Pcs
8	Aluminium	2	M
9	Carbon Fiber	2	M
10	Black Paint 20KG	20	Big Drum
11	Velg	2.3	Pcs
12	Tube	0.4	Pcs
13	Hex Nut 5mm	0.04	Pcs
14	Lock Washer 5mm	0.06	Pcs
15	Socket Head Bolt 5mm	0.03	Pcs

Perencanaan yang diadakan oleh perusahaan ini akan menggunakan 15 macam *raw materials* seperti yang dijabarkan pada tabel 5.3. 15 *raw materials* ini ditentukan dari *bill of material* yang akan diterangkan di tabel 5.4.

- ***Bill of Materials***

Tabel 5.4. *Bill of Materials* Data

No.	Materials Needed	Quantity	UoM	Materials Produced
1	Alumunium	5	m	Deluxe Touring Bike with Black Main Color
2	Tire's Bike	2	Pcs	
3	Chain's Bike	1	Pcs	
4	Handle Bar's Bike	1	Pcs	
5	Seat Kit's Bike	1	Pcs	
6	Pedal's Bike	2	Pcs	
7	Gear's Bike	1	Pcs	
8	Front and Rear Brake's Bike	1	Pcs	
9	Carbon Fiber	5	m	Professional Bike with Black Color
10	Tire's Bike	2	Pcs	
11	Chain's Bike	1	Pcs	

No.	Materials Needed	Quantity	UoM	Materials Produced
12	Handle Bar's Bike	1	Pcs	
13	Seat Kit's Bike	1	Pcs	
14	Pedal's Bike	2	Pcs	
15	Gear's Bike	1	Pcs	
16	Front and Rear Brake's Bike	1	Pcs	

Bill of Materials yang dijabarkan pada tabel 5.4. menjelaskan material-material yang dibutuhkan untuk memproduksi suatu barang yang dibutuhkan pada perusahaan ini.

- ***Resources***

Jumlah pegawai:
- *Direct labour* : 48 orang
- *Indirect labour* : 14 orang

Mesin (2 line):
- *Welding Machine*
- *Laser Cutting Machine*
- *Testing Machine*

- **Perencanaan Keuangan**

Total pembelian aset aktif	: Rp 27.098.691.200,00
Total pembelian *raw materials*	: Rp 46.252.550.550,00
Biaya lain-lain	: Rp 26.648.758.800,00 +
Peminjaman dari Bank	: Rp 100.000.000.000,00
Perencanaan *Net Income*	: Rp 15.187.278.329,60

- **Prosedur Simulasi**
  - **Make to Stock (MTS)**
    Prosedur MTS dapat dilihat pada Gambar 5.1. Penjelasan alur simulasi MTS berdasarkan Gambar 5.1 sebagai berikut:

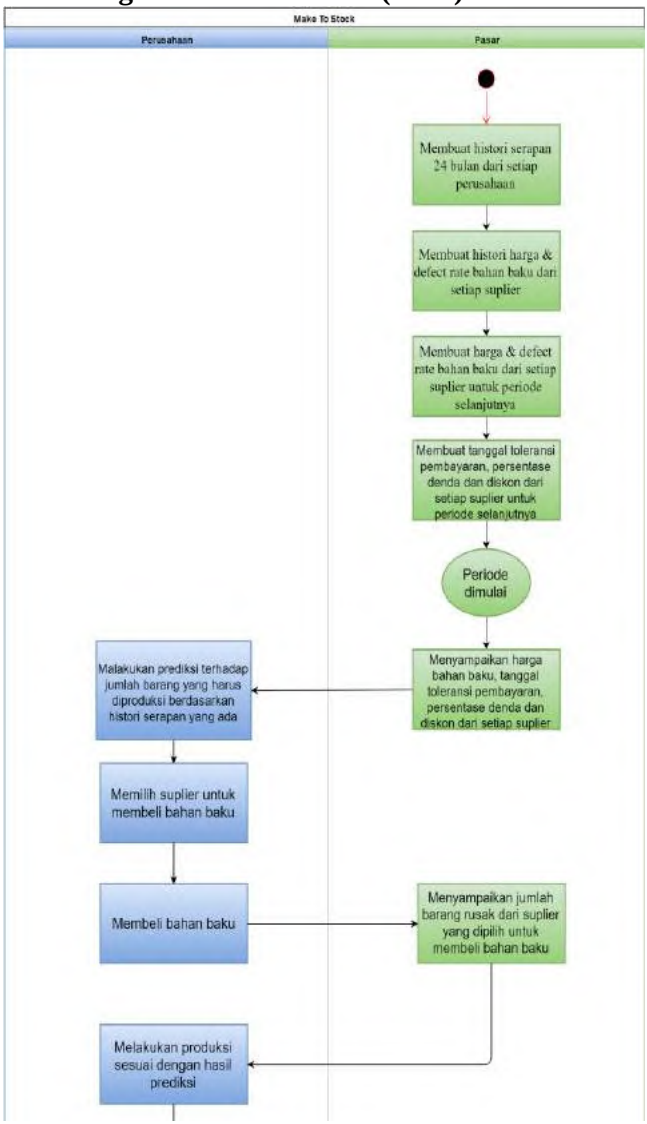
1. Pasar mengeluarkan harga bahan baku, tanggal toleransi pembayaran, persentase denda dan diskon dari setiap supplier.
2. Perusahaan membeli bahan baku dari supplier yang dipilihnya.
3. Pasar menyampaikan jumlah barang rusak dari supplier yang dipilih untuk membeli bahan baku.
4. Perusahaan melakukan produksi berdasarkan hasil forecasting dari histori serapan.
5. Perusahaan memberikan COGS dan biaya iklan kepada pasar yang akan digunakan pasar untuk menentukan serapan.
6. Perusahaan menjual finished goods nya sesuai serapan yang diberikan oleh pasar.

- **Make to Order (MTO)**

Prosedur MTO dapat dilihat pada Gambar 5.2. Penjelasan alur simulasi MTO berdasarkan Gambar 5.2 sebagai berikut:

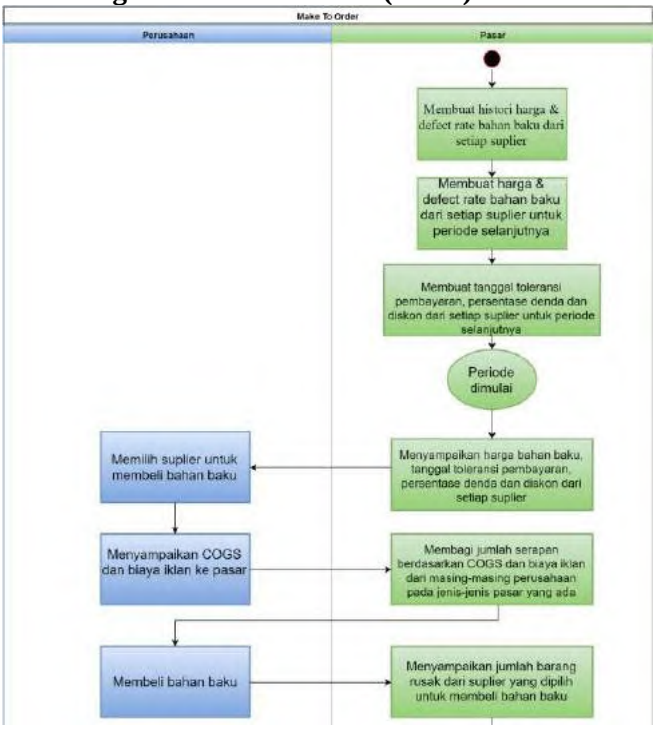
1. Pasar mengeluarkan harga bahan baku, tanggal toleransi pembayaran, persentase denda dan diskon dari setiap supplier.
2. Perusahaan memberikan COGS dan biaya iklan kepada pasar yang akan digunakan pasar untuk menentukan serapan.
3. Perusahaan membeli bahan baku (apabila diperlukan) dari supplier yang dipilihnya.
4. Pasar menyampaikan jumlah barang rusak dari supplier yang dipilih untuk membeli bahan baku.
5. Perusahaan melakukan produksi (apabila diperlukan) hingga finished goods sesuai dengan serapan yang diberikan oleh pasar.
6. Perusahaan menjual finished goods nya sesuai serapan yang diberikan oleh pasar.

- **Alur Diagram Make to Stock (MTS)**



Gambar 5.1. Diagram *Make-to-Stock*

- **Alur Diagram Make to Order (MTO)**



Gambar 5.2. Diagram *Make-to-Order*

- **Aturan Penyerapan Pasar**
  - Rumus random banyaknya penyerapan per hari per produk
    Hi = Ho +((randbetween(4,6)/10)SD)
    Atau
    Hi = Ho -((randbetween(2,4)/10)SD)
    Hi : Jumlah penyerapan hari ini
    Ho : Jumlah penyerapan kemarin
  - Rumus pembagian penyerapan untuk tiap perusahaan per jenis pasar (menggunakan Algoritma Prioritas Wiegers)
    Vpj = Mc BMcHj BHj
    Vpj : Value perusahaan untuk produk p (Deluxe, Profesional) pada jenis pasar j (Grosir, Medium, Retail)
    Mpj : Marketing cost produk p pada jenis pasar j
    BMj : Bobot marketing cost pada jenis pasar j
    Hpj : COGS produk p pada jenis pasar j
    $Pij = Tpj \times Vipji=0nVipj$
    Pij : Jumlah produk (Deluxe, Profesional) yang diserap oleh perusahaan i pada jenis pasar j (Grosir, Medium, Retail)
    Tpj : Total serapan produk p pada jenis pasar j

Vipj: Value perusahaan i untuk produk p pada jenis pasar ji=0nVipj: Total value dari n perusahaan untuk produk p pada jenis pasar j

5.2.1 Pengujian Fungsionalitas Sistem

Pengujian fungsionalitas sistem dilakukan secara mandiri dengan menyiapkan sejumlah skenario sebagai tolak ukur keberhasilan pengujian. Pengujian fungsionalitas dilakukan dengan mengacu pada kasus penggunaan yang telah dijelaskan pada subbab 3.1.6. Pengujian pada kebutuhan fungsionalitas dapat dijabarkan pada subbab berikut.

5.2.1.1 Pengujian Fitur Mengelola *Chart of Accounts*

Pengujian ini terdiri dari pengujian mengelola data *chart of accounts*. Pengujian mengelola data *chart of accounts* yaitu membuat, menyunting, serta menghapus data *account* yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.5.

Tabel 5.5. Pengujian Fitur Mengelola *Chart of Accounts*

ID	UJ.UC.001
Referensi Kasus Penggunaan	UC.001
Nama	Mengelola *Chart of Accounts*
Tujuan Pengujian	Menguji kemampuan sistem dalam mengelola data *Chart of Accounts*
Skenario 1	***Pengguna membuat data Chart of Accounts***
Kondisi Awal	Pengguna berada pada halaman utama *Chart of Accounts*
Data Uji	Inputan data *Chart of Accounts*
Langkah Pengujian	Aktor masuk ke halaman membuat data *Account* baru dan memasukkan Data Uji ke dalam basis data.
Hasil Yang Diharapkan	Data yang dimasukkan ke dalam basis data berhasil dimasukkan.
Hasil Yang Didapat	Data yang dimasukkan tersimpan di basis data

Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman melihat *Account* yang baru saja dibuat
Skenario 2	***Pengguna menyunting data Chart of Accounts.***
Kondisi Awal	Pengguna berada pada halaman utama *Chart of Accounts*
Data Uji	Perubahan data pada *Chart o f Accounts*
Langkah Pengujian	Pengguna memilih menyunting salah satu data *Account* yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman melihat *Account* yang baru saja disunting
Skenario 3	***Aktor menghapus data Account***
Kondisi Awal	Aktor berada pada halaman utama *Chart of Accounts*
Data Uji	-
Langkah Pengujian	Pengguna memilih menghapus salah satu data *Account* yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data
Hasil Yang Didapat	Data berhasil dihapus.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama *Chart of Accounts*

5.2.1.2 Pengujian Fitur Mengelola *Journals*

Pengujian ini terdiri dari pengujian mengelola data *Journals*. Pengujian mengelola data *Journals* yaitu membuat, menyunting,

menghapus, serta menduplikasi data *journals* yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.6.

Tabel 5.6. Pengujian Fitur Mengelola *Journals*

ID	UJ.UC.002
Referensi Kasus Penggunaan	UC.002
Nama	Mengelola *Journals*
Tujuan Pengujian	Menguji kemampuan sistem dalam mengelola data *Journals*
Skenario 1	***Pengguna membuat data Journals***
Kondisi Awal	Pengguna berada pada halaman utama *Journals*
Data Uji	Inputan data *Journals*
Langkah Pengujian	Aktor masuk ke halaman membuat data *Journal* baru dan memasukkan Data Uji ke dalam basis data.
Hasil Yang Diharapkan	Data yang dimasukkan ke dalam basis data berhasil dimasukkan.
Hasil Yang Didapat	Data yang dimasukkan tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman melihat *Journal* yang baru saja dibuat
Skenario 2	***Pengguna menyunting data Journals.***
Kondisi Awal	Pengguna berada pada halaman utama *Journals*
Data Uji	Perubahan data pada *Journals*
Langkah Pengujian	Pengguna memilih menyunting salah satu data *Journal* yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.

Kondisi Akhir	Pengguna berada pada halaman melihat *Journal* yang baru saja disunting
Skenario 3	***Aktor menghapus data Journal***
Kondisi Awal	Aktor berada pada halaman utama *Journals*
Data Uji	-
Langkah Pengujian	Pengguna memilih menghapus salah satu data *Journal* yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data
Hasil Yang Didapat	Data berhasil dihapus.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama *Journals*
Skenario 4	***Aktor menduplikasi data Journal***
Kondisi Awal	Aktor berada pada halaman utama *Journals*
Data Uji	Penduplikasian data pada *Journals*
Langkah Pengujian	Pengguna memilih menduplikasi salah satu data *Journal* yang ada pada sistem, lalu melakukan pembuatan jurnal duplikat dengan mengisi beberapa form yang tersedia dan menyimpannya kembali ke basis data
Hasil Yang Diharapkan	Data yang diduplikat akan masuk ke dalam basis data
Hasil Yang Didapat	Data berhasil diduplikat.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman melihat *Journal* yang baru saja diduplikat.

5.2.1.3 Pengujian Fitur Mengelola *Period*

Pengujian ini terdiri dari pengujian mengelola data *period*. Pengujian mengelola data *period* yaitu membuat, menyunting, serta

menghapus data *period* yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.7.

Tabel 5.7. Pengujian Fitur Mengelola *Period*

ID	UJ.UC.003
Referensi Kasus Penggunaan	UC.003
Nama	Mengelola *Period*
Tujuan Pengujian	Menguji kemampuan sistem dalam mengelola data *Period*
Skenario 1	***Pengguna membuat data Period***
Kondisi Awal	Pengguna berada pada halaman utama *Period*
Data Uji	Inputan data *Period*
Langkah Pengujian	Aktor masuk ke halaman membuat data *Period* baru dan memasukkan Data Uji ke dalam basis data.
Hasil Yang Diharapkan	Data yang dimasukkan ke dalam basis data berhasil dimasukkan.
Hasil Yang Didapat	Data yang dimasukkan tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman melihat *Period* yang baru saja dibuat
Skenario 2	***Pengguna menyunting data Period.***
Kondisi Awal	Pengguna berada pada halaman utama *Period*
Data Uji	Perubahan data pada *Period*
Langkah Pengujian	Pengguna memilih menyunting salah satu data *Period* yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.

Kondisi Akhir	Pengguna berada pada halaman melihat *Period* yang baru saja disunting
Skenario 3	***Aktor menghapus data Period***
Kondisi Awal	Aktor berada pada halaman utama *Period*
Data Uji	-
Langkah Pengujian	Pengguna memilih menghapus salah satu data *Period* yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data
Hasil Yang Didapat	Data berhasil dihapus.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Period

5.2.1.4 Pengujian Fitur Melihat Data *Chart of Accounts*

Pengujian ini terdiri dari pengujian melihat data *chart of accounts*. Pengujian melihat data *chart of accounts* yaitu melihat data *account* yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.8.

Tabel 5.8. Pengujian Fitur Melihat Data *Chart of Accounts*

ID	UJ.UC.004
Referensi Kasus Penggunaan	UC.004
Nama	Pengujian Fitur Melihat Data *Chart of Accounts*
Tujuan Pengujian	Menguji kemampuan aplikasi melihat data *chart of accounts*
Skenario 1	***Pengguna melihat data chart of accounts***
Kondisi Awal	Aktor berada pada halaman utama modul *General Ledger*
Data Uji	-
Langkah Pengujian	Aktor masuk ke halaman submodul *Chart of Accounts*

Hasil Yang Diharapkan	Tertampil semua data *account* yang tersimpan di basis data
Hasil Yang Didapat	Tertampil semua data *account* yang tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman utama submodul *Chart of Accounts*

5.2.1.5 Pengujian Fitur Melihat Data *Account Balance*

Pengujian ini terdiri dari pengujian melihat data *Account Balance*. Pengujian melihat data *account balance* yaitu melihat data debit, kredit, serta *balance* dari masing-masing *account* yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.9.

Tabel 5.9. Pengujian Fitur Melihat Data *Account Balance*

ID	UJ.UC.005
Referensi Kasus Penggunaan	UC.005
Nama	Pengujian Fitur Melihat Data *Account Balance*
Tujuan Pengujian	Menguji kemampuan aplikasi melihat data *account balance*
Skenario 1	***Pengguna melihat data account balance***
Kondisi Awal	Aktor berada pada halaman utama modul *General Ledger*
Data Uji	Inputan data *Chart of Accounts* dan *Journals*
Langkah Pengujian	Aktor masuk ke halaman melihat data secara keseluruhan
Hasil Yang Diharapkan	Tertampil detail data dari semua *account* beserta debit, kredit, dan *balance*nya
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan

5.2.1.6 Pengujian Fitur Melihat Data *Period*

Pengujian ini terdiri dari pengujian melihat data *Period*. Pengujian melihat data *period* yaitu melihat data *financial period* yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.10.

Tabel 5.10. Pengujian Fitur Melihat Data *Period*

ID	UJ.UC.006
Referensi Kasus Penggunaan	UC.006
Nama	Pengujian Fitur Melihat Data *Period*
Tujuan Pengujian	Menguji kemampuan aplikasi melihat data *period*
Skenario 1	***Pengguna melihat data period***
Kondisi Awal	Aktor berada pada halaman utama modul *General Ledger*
Data Uji	-
Langkah Pengujian	Aktor masuk ke halaman submodul *Period*
Hasil Yang Diharapkan	Tertampil semua data *period* yang tersimpan di basis data
Hasil Yang Didapat	Tertampil semua data *period* yang tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman utama submodul *Period*

5.2.1.7 Pengujian Fitur Melihat Data *Journals*

Pengujian ini terdiri dari pengujian melihat data *Journals*. Pengujian melihat data *journals* yaitu melihat data *journal* yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.11.

Tabel 5.11. Pengujian Fitur Melihat Data *Journals*

ID	UJ.UC.007
Referensi Kasus Penggunaan	UC.007
Nama	Pengujian Fitur Melihat Data *Journals*

Tujuan Pengujian	Menguji kemampuan aplikasi melihat data *journals*
Skenario 1	***Pengguna melihat data journals***
Kondisi Awal	Aktor berada pada halaman utama modul *General Ledger*
Data Uji	-
Langkah Pengujian	Aktor masuk ke halaman submodul *Journals*
Hasil Yang Diharapkan	Tertampil semua data *journal* yang tersimpan di basis data
Hasil Yang Didapat	Tertampil semua data *journal* yang tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman utama submodul *Journals*

5.2.1.8 Pengujian Fitur Melihat Detail *Chart of Accounts*

Pengujian ini terdiri dari pengujian melihat detail *Chart of Accounts*. Pengujian melihat detail *chart of accounts* yaitu melihat data *account* secara terperinci yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.12.

Tabel 5.12. Pengujian Fitur Melihat Detail *Chart of Accounts*

ID	UJ.UC.008
Referensi Kasus Penggunaan	UC.008
Nama	Pengujian Fitur Melihat Detail *Chart of Accounts*
Tujuan Pengujian	Menguji kemampuan aplikasi melihat data *chart of accounts* secara terperinci
Skenario 1	***Pengguna melihat detail chart of accounts***
Kondisi Awal	Aktor berada pada halaman utama *Chart of Accounts*
Data Uji	-
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari *account* pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail

Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan

5.2.1.9 Pengujian Fitur Melihat Detail *Period*

Pengujian ini terdiri dari pengujian melihat detail *Period*. Pengujian melihat detail *period* yaitu melihat data *period* secara terperinci yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.13.

Tabel 5.13. Pengujian Fitur Melihat Detail *Period*

ID	UJ.UC.009
Referensi Kasus Penggunaan	UC.009
Nama	Pengujian Fitur Melihat Detail *Period*
Tujuan Pengujian	Menguji kemampuan aplikasi melihat data *period* secara terperinci
Skenario 1	***Pengguna melihat detail period***
Kondisi Awal	Aktor berada pada halaman utama *Period*
Data Uji	-
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari *period* pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan

5.2.1.10 Pengujian Fitur Melihat Detail *Journals*

Pengujian ini terdiri dari pengujian melihat detail *Journals*. Pengujian melihat detail *journal* yaitu melihat data *journal* secara terperinci yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.14.

Tabel 5.14. Pengujian Fitur Melihat Detail *Journals*

ID	UJ.UC.010
Referensi Kasus Penggunaan	UC.010
Nama	Pengujian Fitur Melihat Detail *Journals*
Tujuan Pengujian	Menguji kemampuan aplikasi melihat data *journal* secara terperinci
Skenario 1	***Pengguna melihat detail journal***
Kondisi Awal	Aktor berada pada halaman utama *Journals*
Data Uji	-
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari *journal* pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan

5.2.1.11 Pengujian Fitur Melihat Laporan *Balance Sheet*

Pengujian ini terdiri dari pengujian melihat laporan *Balance Sheet*. Pengujian melihat laporan *balance sheet* yaitu melihat laporan *balance sheet* yang berdasar pada jurnal, akun, serta periode yang telah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.15.

Tabel 5.15. Pengujian Fitur Melihat Laporan *Balance Sheet*

ID	UJ.UC.011
Referensi Kasus Penggunaan	UC.011
Nama	Pengujian Fitur Melihat Laporan *Balance Sheet*
Tujuan Pengujian	Menguji kemampuan aplikasi melihat laporan *balance sheet*
Skenario 1	***Pengguna melihat laporan balance sheet***
Kondisi Awal	Aktor berada pada halaman utama modul *General Ledger*
Data Uji	Inputan data *Journals*, *Period*, dan *Chart of Accounts*

Langkah Pengujian	Aktor masuk ke halaman melihat laporan *Balance Sheet* dan memilih tipe laporan yang akan ditampilkan
Hasil Yang Diharapkan	Tertampil laporan *balance sheet*
Hasil Yang Didapat	Laporan *balance sheet* dapat ditampilkan oleh sistem
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman laporan *balance sheet* sesuai tipe yang dipilih

5.2.1.12 Pengujian Fitur Melihat Laporan *Income Statement*

Pengujian ini terdiri dari pengujian melihat laporan *Income Statement*. Pengujian melihat laporan *income statement* yaitu melihat laporan *income statement* yang berdasar pada jurnal, akun, serta periode yang telah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.16.

Tabel 5.16. Pengujian Fitur Melihat Laporan *Income Statement*

ID	UJ.UC.012
Referensi Kasus Penggunaan	UC.012
Nama	Pengujian Fitur Melihat Laporan *Income Statement*
Tujuan Pengujian	Menguji kemampuan aplikasi melihat laporan *income statement*
Skenario 1	***Pengguna melihat laporan income statement***
Kondisi Awal	Aktor berada pada halaman utama modul *General Ledger*
Data Uji	Inputan data *Journals*, *Period*, dan *Chart of Accounts*
Langkah Pengujian	Aktor masuk ke halaman melihat laporan *Income Statement* dan memilih tipe laporan yang akan ditampilkan
Hasil Yang Diharapkan	Tertampil laporan *income statement*
Hasil Yang Didapat	Laporan *income statement* dapat ditampilkan oleh sistem
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman laporan *income statement* sesuai tipe yang dipilih

5.2.1.13 Pengujian Fitur Melihat Laporan *Cash Flow*

Pengujian ini terdiri dari pengujian melihat laporan *Cash Flow*. Pengujian melihat laporan *cash flow* yaitu melihat laporan *cash flow* yang berdasar pada jurnal, akun, serta periode yang telah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.17.

Tabel 5.17. Pengujian Fitur Melihat Laporan *Cash Flow*

ID	UJ.UC.013
Referensi Kasus Penggunaan	UC.013
Nama	Pengujian Fitur Melihat Laporan *Cash Flow*
Tujuan Pengujian	Menguji kemampuan aplikasi melihat laporan *cash flow*
Skenario 1	***Pengguna melihat laporan cash flow***
Kondisi Awal	Aktor berada pada halaman utama modul *General Ledger*
Data Uji	Inputan data *Journals*, *Period*, dan *Chart of Accounts*
Langkah Pengujian	Aktor masuk ke halaman melihat laporan *Cash Flow* dan memilih tipe laporan yang akan ditampilkan
Hasil Yang Diharapkan	Tertampil laporan *cash flow*
Hasil Yang Didapat	Laporan *cash flow* dapat ditampilkan oleh sistem
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman laporan *cash flow* sesuai tipe yang dipilih

5.2.2 Pengujian Fungsional *Role Based Access Control* (RBAC)

Pengujian ini terdiri dari pengujian mengela *Role Based Access Control (RBAC)*. Rincian skenario pengujian pada kasus penggunaan pengujian mengelola *Role Based Access Control (RBAC)* dapat dilihat pada Tabel 5.18.

Tabel 5.18. Pengujian Fitur Mengelola *Role Based Access Control*

ID	UJ.UC.RBAC
Referensi Kasus Penggunaan	UC. RBAC
Nama	Pengujian mengelola *Role Based Access Control (RBAC).*
Tujuan Pengujian	Menguji kemampuan aplikasi mengelola data *Role Based Access Control (RBAC).*
Skenario 1	***Pengguna membuat user baru.***
Kondisi Awal	Pengguna berada pada halaman dashboard admin.
Data Uji	Inputan data *user* baru:
Langkah Pengujian	Pengguna memilih kegiatan *Add New User*, lalu memasukkan data inputan dan menekan tombol "*Add User*".
Hasil Yang Diharapkan	*User* yang baru terdapat pada halaman daftar *user*.
Hasil Yang Didapat	*User* yang baru terdapat pada halaman daftar *user*.
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman halaman daftar *user* dengan tambahan data *user* yang baru.
Skenario 2	***Pengguna menyunting data user tertentu.***
Kondisi Awal	Pengguna berada pada halaman daftar *user*.
Data Uji	Perubahan data *user*:
Langkah Pengujian	Pengguna memilih kegiatan menyunting *user* tertentu, lalu melakukan perubahan data dan menekan tombol "*Update*".
Hasil Yang Diharapkan	*User* yang disunting mengalami perubahan data sesuai inputan.
Hasil Yang Didapat	*User* yang disunting mengalami perubahan data sesuai inputan.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman daftar user.
Skenario 3	***Pengguna menghapus user tertentu.***

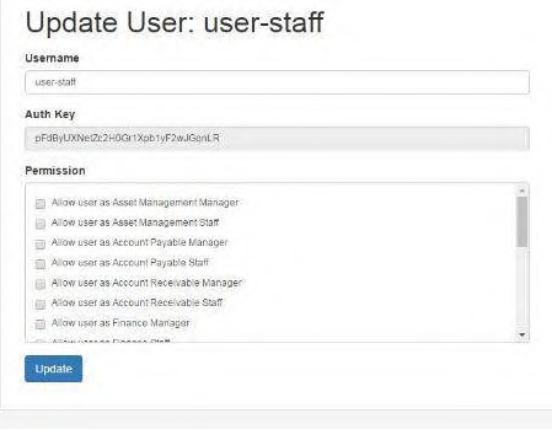
Kondisi Awal	Pengguna berada pada halaman daftar user.
Data Uji	-
Langkah Pengujian	Pengguna memilih kegiatan menghapus kegiatan tertentu, lalu menekan tombol "*Yes*" untuk konfirmasi penghapusan.
Hasil Yang Diharapkan	*User* yang baru dihapus tidak tampil pada daftar *user*.
Hasil Yang Didapat	*User* yang baru dihapus tidak tampil pada daftar *user*.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman daftar *user*.

Gambar 5.3 menunjukkan proses login oleh admin. Gambar 5.4 menunjukkan hasil setelah proses login oleh admin. Gambar 5.5 menunjukkan proses penginputan data *user* untuk skenario 1. Sedangkan untuk Kemudian pada Gambar 5.6 menunjukkan proses penyuntingan data *user* untuk skenario 2. Sedangkan pada Gambar 5.7 dapat dilihat proses penghapusan *user* untuk skenario 3. Dengan melihat hasil pengujian pada ketiga skenario diatas, bisa disimpulkan bahwa Kasus Penggunaan UC.RBAC telah bekerja dengan baik seperti yang diharapkan.

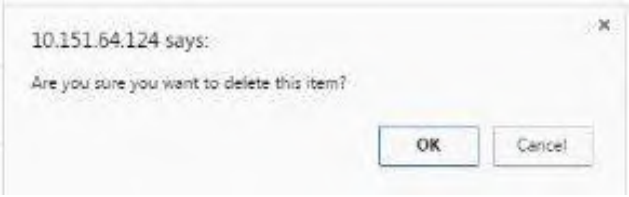
Gambar 5.3. Proses Login Admin

Gambar 5.4. Tampilan Awal Setelah Login Berhasil Dilakukan

Gambar 5.5. Proses Admin Menambahkan User Baru



Gambar 5.6. Proses Admin Menyunting Data User



Gambar 5.7. Proses Admin Menghapus Data User

5.2.3 Pengujian Fungsional Basis Data Terdistribusi

Pengujian Basis Data Terdistribusi (BDT) dilakukan untuk menguji 2 aspek utama dalam konsep BDT, yaitu replikasi dan *high availability*.

Sistem BDT menggunakan 4 server fisik. Dengan rincian sebagai berikut:

1. *Server* aplikasi, adalah server yang di dalamnya terdapat *web server* beserta file aplikasi ERP.
2. *Management Node*, adalah *server* yang berfungsi sebagai pusat pengaturan sistem BDT.
3. *Data Node*, adalah 2 buah server penyimpan data.

- **Prosedur Simulasi**

  Replikasi

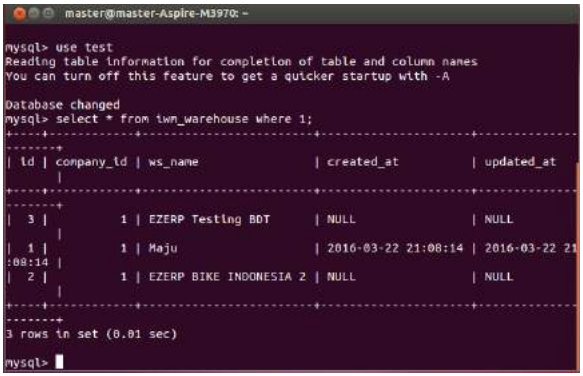
  1. Memastikan seluruh *server* dan sistem BDT dalam keadaan *online*.
  2. Melakukan penambahan. perubahan, dan penghapusan data melalui aplikasi ERP yang ditunjukkan pada Gambar 5.8.
  3. Melakukan pengecekan terhadap hasil penambahan, perubahan, dan penghapusan data pada seluruh *data node* yang ditampilkan pada Gambar 5.9. dan Gambar 5.10.

  High Availability

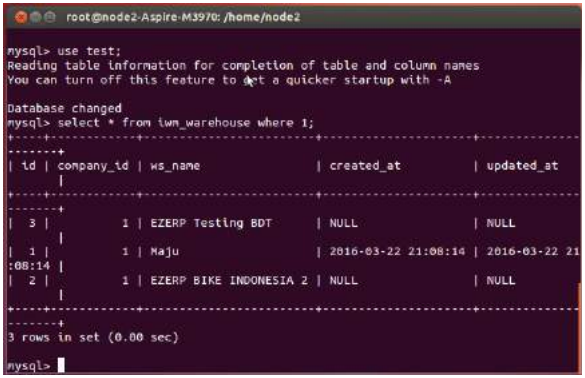
  1. Mematikan *server* aplikasi, *management server* dan sistem BDT pada salah satu *node.* Ditampilkan pada Gambar 5.12

2. Melakukan penambahan, perubahan, dan penghapusan data melalui aplikasi yang ditunjukkan pada Gambar 5.13
3. Melakukan pengecekan terhadap hasil pemrosesan data pada *node* yang masih aktif yang ditampilkan pada Gambar 5.14.

Gambar 5.8 Pengujian Fitur Replikasi pada Sistem



Gambar 5.9. Pengujian Fitur Replikasi pada Database Server 1

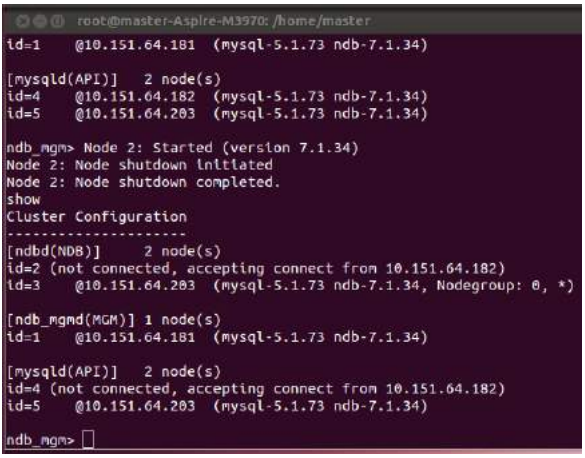


Gambar 5.10. Pengujian Fitur Replikasi pada Database Server 2

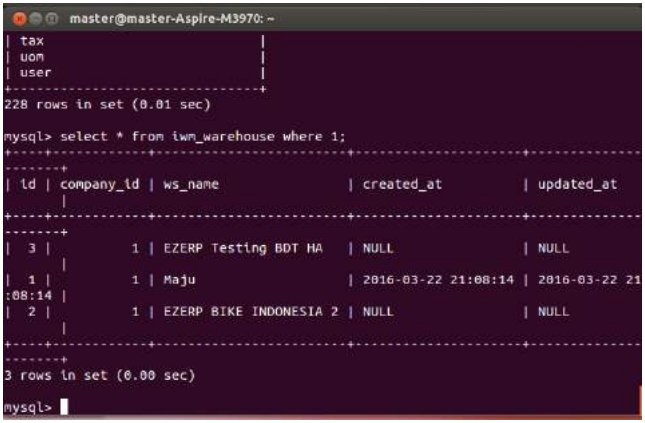
Warehouse 3

ID	3
Company	EZTENANT BIKE INDONESIA
Warehouse Name	EZERP Testing BDT HA

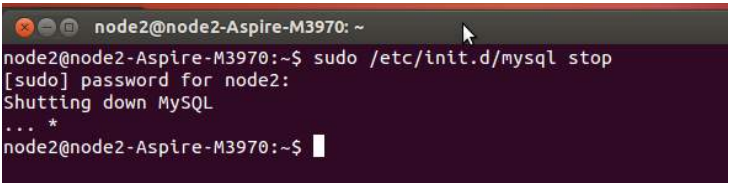
Gambar 5.11. Pengujian Fitur *High-Availability* pada Sistem



Gambar 5.12. Pengecekan *Availability* pada node *Server*



Gambar 5.13. Pengujian Fitur *High-Availability* pada Database *Server* 1



Gambar 5.14. Pengujian Fitur *High-Availability* pada Database Server 2

5.3 Evaluasi Pengujian

Pada subbab ini akan diberikan hasil evaluasi dari pengujian-pengujian yang telah dilakukan. Evaluasi yang diberikan meliputi evaluasi pengujian kebutuhan fungsional dan evaluasi pengujian kegunaan.

5.3.1 Evaluasi Pengujian Fungsionalitas Sistem

Rangkuman mengenai hasil pengujian fungsionalitas sistem dapat dilihat pada Tabel 5.19. Berdasarkan data pada tabel tersebut, semua skenario pengujian berhasil dan program berjalan dengan baik. Sehingga bisa ditarik disimpulkan bahwa fungsionalitas dari program telah bisa bekerja sesuai dengan yang diharapkan.

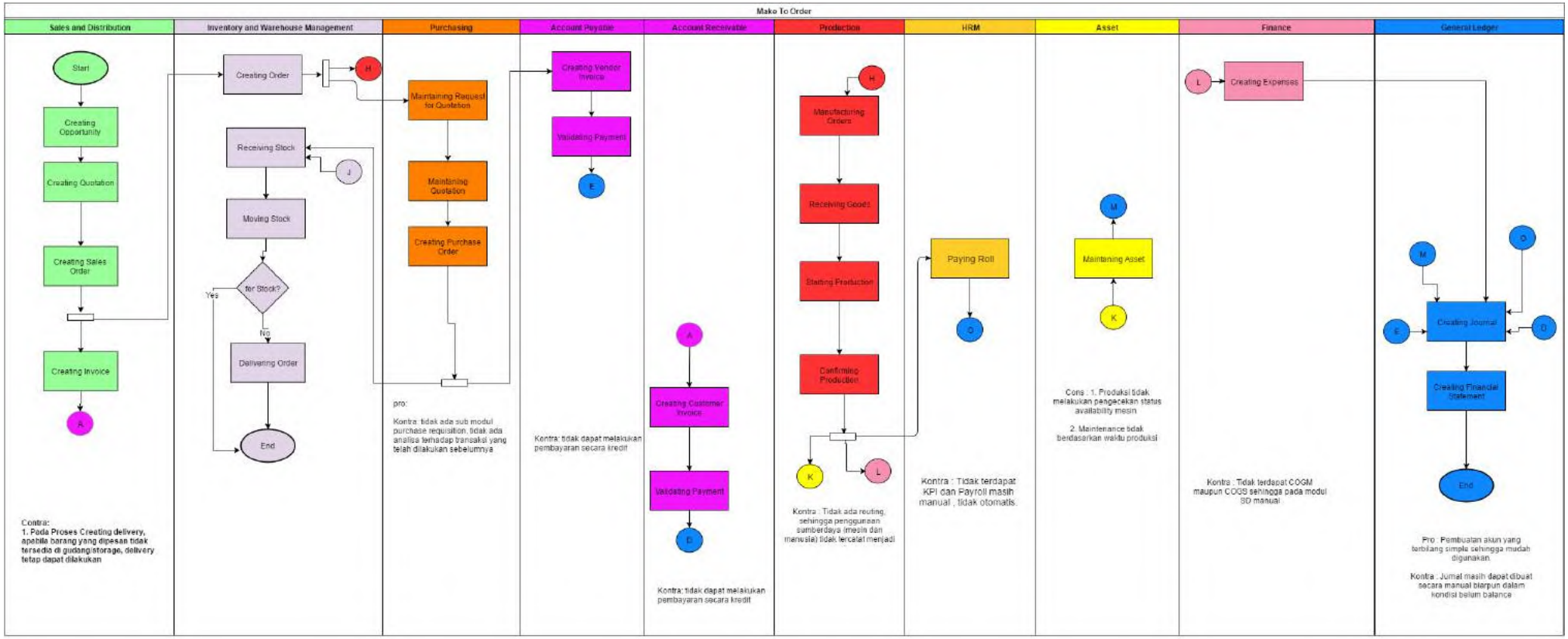
Tabel 5.19. Evaluasi Pengujian Fungsionalitas Sistem

No.	ID	Nama Kasus Penggunaan	Skenario	Hasil
1	UJ.UC.001	Mengelola *Chart of Accounts*	Skenario 1	Berhasil
			Skenario 2	Berhasil
			Skenario 3	Berhasil
2	UJ.UC.002	Mengelola *Journals*	Skenario 1	Berhasil
			Skenario 2	Berhasil
			Skenario 3	Berhasil
3	UJ.UC.003	Mengelola *Period*	Skenario 1	Berhasil
			Skenario 2	Berhasil
			Skenario 3	Berhasil
4	UJ.UC.004	Melihat Data *Chart of Accounts*	Skenario 1	Berhasil

5	UJ.UC.005	Melihat Data *Account Balance*	Skenario 1	Berhasil
6	UJ.UC.006	Melihat Data *Period*	Skenario 1	Berhasil
7	UJ.UC.007	Melihat Data *Journals*	Skenario 1	Berhasil
8	UJ.UC.008	Melihat Detail *Chart of Accounts*	Skenario 1	Berhasil
9	UJ.UC.009	Melihat Detail *Period*	Skenario 1	Berhasil
10	UJ.UC.010	Melihat Detail *Journals*	Skenario 1	Berhasil
11	UJ.UC.011	Melihat Laporan *Balance Sheet*	Skenario 1	Berhasil
12	UJ.UC.012	Melihat Laporan *Income Statement*	Skenario 1	Berhasil
13	UJ.UC.013	Melihat Laporan *Cash Flow*	Skenario 1	Berhasil

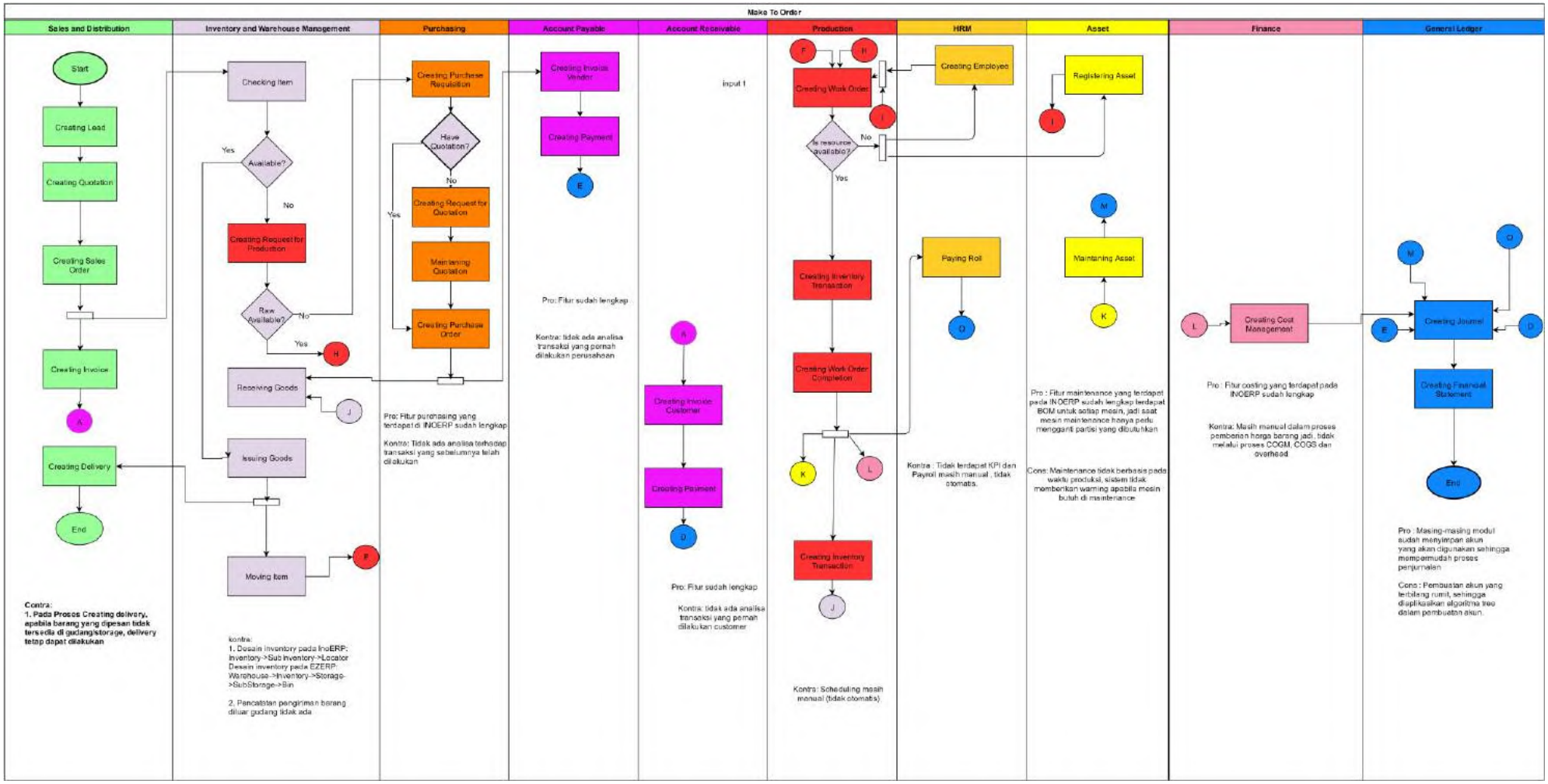
LAMPIRAN A – PROSES BISNIS

A.1 Proses Bisnis Odoo



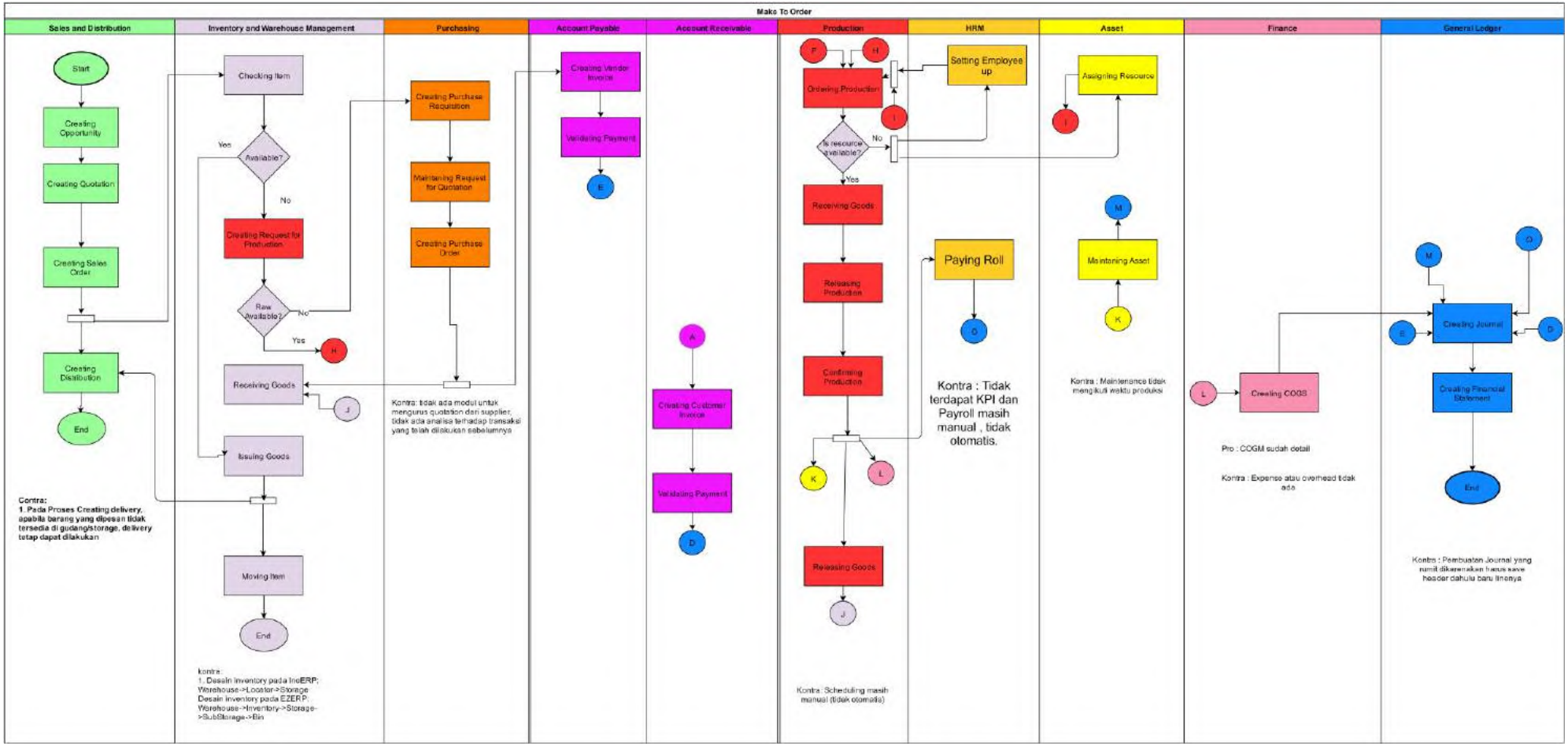
Gambar A.1. Diagram Proses Bisnis Odoo

A.2. Proses Bisnis InoERP



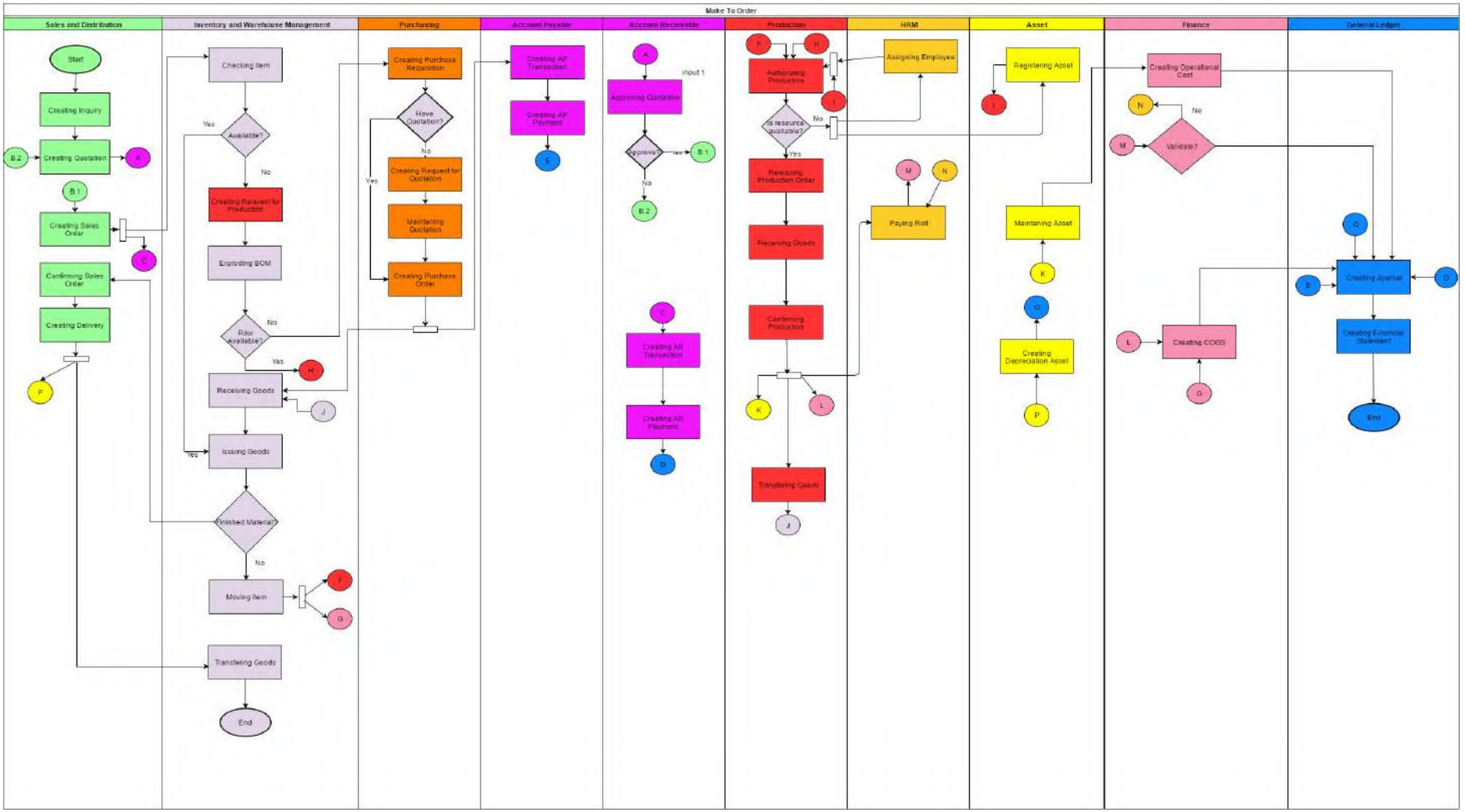
Gambar A.2. Proses Bisnis InoERP

A.3. Proses Bisnis Adempiere



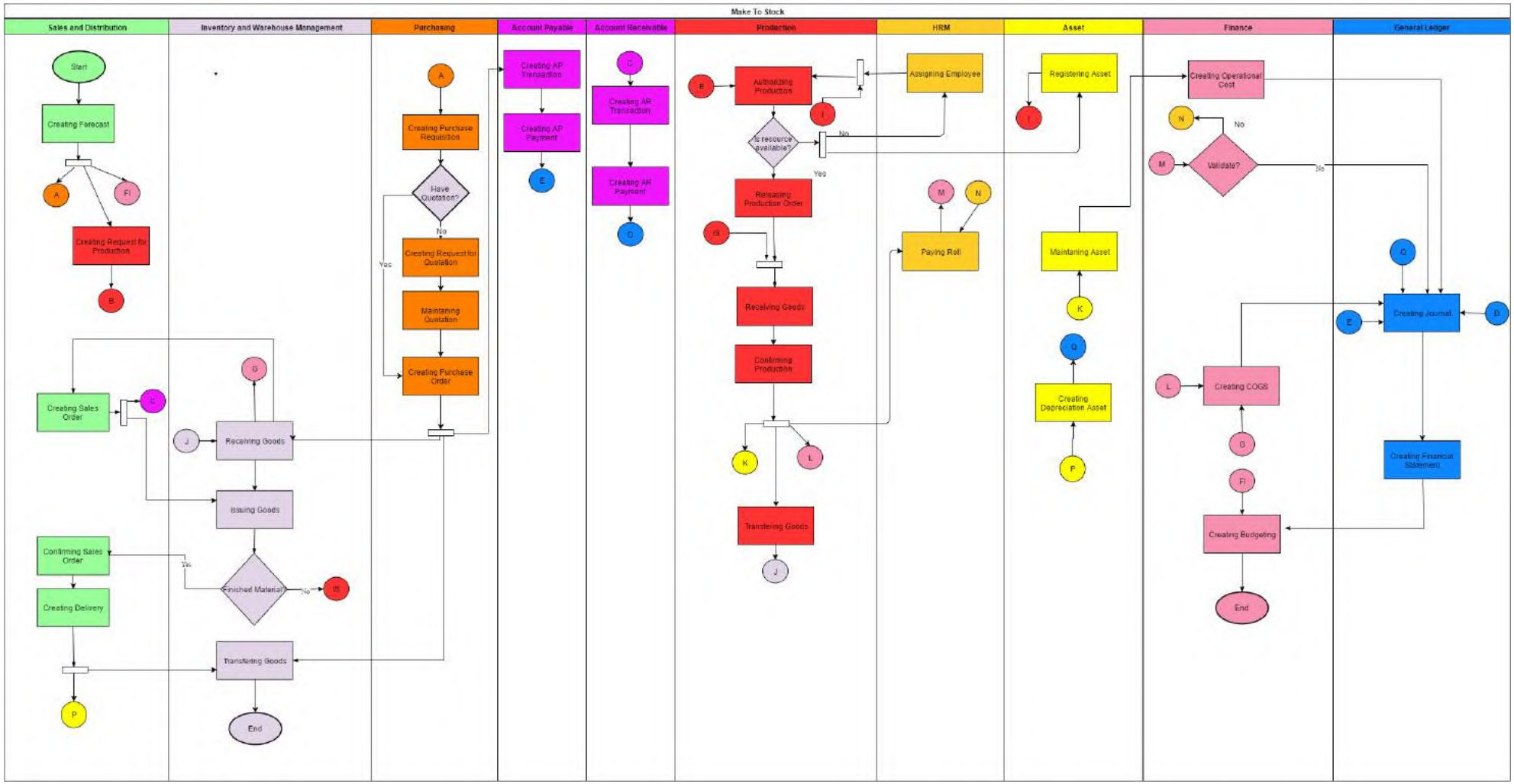
Gambar A.3. Proses Bisnis Adempiere

A.4. Proses Bisnis *Make-to-Order* ERP 2016



Gambar A.4. Proses Bisnis *Make-to-Order* ERP 2016

A.5. Proses Bisnis *Make-to-Stock* ERP 2016



Gambar A.5. Proses Bisnis *Make-to-Stock* ERP 2016

[Halaman ini sengaja dikosongkan]

LAMPIRAN B – KODE SUMBER

B.1. Kode Sumber Halaman Utama Modul

Kode sumber melihat halaman utama modul berfungsi untuk menampilkan ringkasan dari hal-hal penting berkaitan dengan *General Ledger*. Pengaturan melihat kode sumber halaman utama modul diatur pada fungsi `actionIndex()` yang ditunjukkan pada Kode Sumber B.1.

```
public function actionIndex($liq = 1, $act = 1)
{
$new_date = new \DateTime('NOW');
$new_date->modify('-20 day');
$date = $new_date->format('Y-m-d');

$date2 = date('Y-m-d');

$qDate = "SELECT gl_period.period_name as dates FROM
        gl_period WHERE period_name >= '$date' AND
        period_name <= '$date2' ORDER BY period_name ASC";
$date = $db->createCommand($qDate)->queryAll();
$profitChart = array();
foreach ($date as $key => $value) {
        $profitChart[$count]['profit'] = 0;
        $tgl = date('d F Y', strtotime($value['dates']));
        $profitChart[$count]['date'] = $tgl;
        $models = IstatementController::actionCari(0,
        $value['dates'], 3, 0);
        $mark = 0;
    foreach ($models as $key1 => $value1) {
        if(!isset($value1['balance'])) $value1['balance']
        = 0;

        if($value1['level'] == 1 && $mark != 1) {
            $profitChart[$count]['profit'] +=
        $value1['balance'];
            $mark++;
        }
        else if($value1['level'] == 1 && $mark == 1) {
            $profitChart[$count]['profit'] -=
        $value1['balance'];
            $mark++;
        }
    }
    $count++;
```

```
}
return $this->render('index', [
     'profitChart' => $profitChart,
         ]);
}
```

Kode Sumber B.1. Kode Sumber Halaman Utama Modul

Pada Kode Sumber B.1, ditunjukkan potongan kode sumber untuk halaman utama modul *General Ledger*. `actionIndex()` adalah fungsi yang digunakan untuk menampilkan data-data yang dibutuhkan yang tersimpan di dalam basis data untuk ditampilkan ke halaman utama modul *General Ledger*.

B.2. Kode Sumber Melihat Daftar Data Submodul

Kode sumber melihat daftar data submodul berfungsi untuk menampilkan daftar data submodul yang terdapat pada modul *General Ledger*. Pengaturan melihat daftar data submodul diatur pada fungsi `actionIndex()` yang ditunjukkan pada Kode Sumber B.2.

```
public function actionIndex()
{
    $searchModel = new PeriodSearch();
    $dataProvider     =     $searchModel->search(Yii::$app-
    >request->queryParams);

    return $this->render('index', [
        'searchModel' => $searchModel,
        'dataProvider' => $dataProvider,
    ]);
}
```

Kode Sumber B.2. Kode Sumber Melihat Daftar Data Submodul

Pada Kode Sumber B.2, ditunjukkan potongan kode sumber untuk melihat daftar data submodul yang terdapat pada modul *General Ledger*. `actionIndex()` adalah fungsi yang digunakan untuk menampilkan daftar data submodul yang dibutuhkan yang tersimpan di dalam basis data untuk ditampilkan ke halaman daftar data submodul.

B.3. Kode Sumber Melihat Detail Data Submodul

Kode sumber melihat detail data submodul berfungsi untuk menampilkan detail data submodul yang terdapat pada modul *General Ledger*. Pengaturan melihat detail data submodul diatur pada fungsi `actionView()` yang ditunjukkan pada Kode Sumber B.3.

```
public function actionView($id)
{
    $model = $this->findModel($id);
    return $this->render('view', [
        'journal' => $results,
        'model' => $this->findModel($id),
    ]);
  }
```

Kode Sumber B.3. Kode Sumber Melihat Detail Data Submodul

Pada Kode Sumber B.3, ditunjukkan potongan kode sumber untuk melihat detail data submodul yang terdapat pada modul *General Ledger*. `actionView()` adalah fungsi yang digunakan untuk menampilkan detail data submodul yang dibutuhkan yang tersimpan di dalam basis data untuk ditampilkan ke halaman detail data submodul.

B.4. Kode Sumber Menambah Data Submodul

Kode sumber menambah data submodul berfungsi untuk menambahkan suatu data submodul yang terdapat pada modul *General Ledger* ke dalam database terkait. Pengaturan menambah data submodul diatur pada fungsi `actionCreate()` yang ditunjukkan pada Kode Sumber B.4.

```
public function actionCreate()
{
    $model = new Period();
    if   ($model->loadAll(Yii::$app->request->post())   &&
    $model->saveAll()) {
        return  $this->redirect(['view',  'id'  =>  $model-
    >id]);
    } else {
        return $this->render('create', [
            'model' => $model,
        ]);
```

```
    }
}
```

Kode Sumber B.4. Kode Sumber Menambah Data Submodul

Pada Kode Sumber B.4, ditunjukkan potongan kode sumber untuk menambah data submodul yang terdapat pada modul *General Ledger*. `actionCreate()` adalah fungsi yang digunakan untuk menambahkan data submodul ke dalam basis data, yang kemudian ditampilkan ke halaman detail data submodul.

B.5. Kode Sumber Menyunting Data Submodul

Kode sumber menyunting data submodul berfungsi untuk menyunting suatu data pada submodul yang terdapat pada modul *General Ledger*. Pengaturan menyunting data submodul diatur pada fungsi `actionUpdate()` yang ditunjukkan pada Kode Sumber B.5.

```
public function actionUpdate($id)
{
    $model = $this->findModel($id);
    if    ($model->loadAll(Yii::$app->request->post())    &&
     $model->saveAll())
    {
        return  $this->redirect(['view',  'id'  =>  $model-
     >id]);


    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}
```

Kode Sumber B.5. Kode Sumber Menyunting Data Submodul

Pada Kode Sumber B.5, ditunjukkan potongan kode sumber untuk menyunting data submodul yang terdapat pada modul *General Ledger*. `actionUpdate()` adalah fungsi yang digunakan untuk menyunting data submodul yang ada di dalam

basis data, yang kemudian ditampilkan ke halaman detail data submodul.

B.6. Kode Sumber Menghapus Data Submodul

Kode sumber menghapus data submodul berfungsi untuk menghapus suatu data submodul yang terdapat pada modul *General Ledger*. Pengaturan menghapus data submodul diatur pada fungsi `actionDelete()` yang ditunjukkan pada Kode Sumber B.6.

```
public function actionDelete($id)
{
    $this->findModel($id)->deleteWithRelated();
    return $this->redirect(['index']);
}
```

Kode Sumber B.6. Kode Sumber Menghapus Data Submodul

Pada Kode Sumber B.6, ditunjukkan potongan kode sumber untuk menghapus data submodul yang terdapat pada modul *Sales and Distribution*. `actionDelete()` adalah fungsi yang digunakan untuk menghapus data submodul yang tersimpan di dalam basis data untuk ditampilkan ke halaman daftar data submodul.

B.7. Kode Sumber Melihat Tipe Laporan

Kode sumber melihat tipe laporan berfungsi untuk memilih tipe laporan yang akan ditampilkan. Pengaturan melihat tipe laporan diatur pada fungsi `actionIndexajax()` yang ditunjukkan pada Kode Sumber B.7.

```
public function actionIndexajax(){
    return $this->renderAjax('index');
}
```

Kode Sumber B.7. Kode Sumber Melihat Tipe Laporan

Pada Kode Sumber B.7, ditunjukkan potongan kode sumber untuk menampilkan tipe laporan yang akan dicetak, bisa berupa

tahunan, bulanan, harian, maupun 3 bulanan. `actionIndexajax()` adalah fungsi yang digunakan untuk menampilkan pop-up berupa tipe laporan yang bisa dicetak.

B.8. Kode Sumber Melihat Laporan

Kode sumber melihat laporan berfungsi untuk menampilkan laporan keuangan yang terdapat pada modul *General Ledger*. Pengaturan melihat laporan ditunjukkan pada Kode Sumber B.8, Kode Sumber B.9, dan Kode Sumber B.10.

```
public function actionYearly($year = NULL){

    $queryYear          =          "SELECT          DISTINCT
     YEAR(gl_period.period_name) as year from gl_period";
    $db = Yii::$app->db;
    $dataYear = $db->createCommand($queryYear)->queryAll();

    $queryLast          =          "SELECT          DISTINCT
     MAX(gl_period.period_name) as day from gl_period where
     YEAR(gl_period.period_name) = $year";
    $lastDate = $db->createCommand($queryLast)->queryOne();
    $date = date('d F Y', strtotime($lastDate['day']));
    $queryCoa   =   "SELECT   gl_group.coa_name   as   name,
     gl_group.id from gl_group ORDER BY id ASC";
    $coaList = $db->createCommand($queryCoa)->queryAll();

    $models = $this->actionBalance(0, $year, 0, 0);
    foreach ($models as $key => $value) {
        if($value['level'] == 1 && $mark == 0) {
            $balance += $value['balance'];
            $mark++;
        }
        else if($value['level'] == 1 && $mark == 1) {
            $balance -= $value['balance'];
            $le += $value['balance'];
        }
    }
    return $this->render('yearly', [
         'date' => $date,
         'models' => $models,
         'dataYear' => $dataYear,
         'year' => $year,
         'coaList' => $coaList,
         'balance' => $balance,
         'lastBalance' => $lastBalance,
         'le' => $le,
```

```
            'leOld' => $leOld,
        ]);
}
```

Kode Sumber B.8. Kode Sumber Melihat Laporan *Balance Sheet*

```
public function actionYearly($year = NULL)
{
    $queryYear          =          "SELECT          DISTINCT
     YEAR(gl_period.period_name) as year from gl_period";
    $db = Yii::$app->db;
    $dataYear = $db->createCommand($queryYear)->queryAll();

    $queryLast          =          "SELECT          DISTINCT
     MAX(gl_period.period_name) as day from gl_period where
     YEAR(gl_period.period_name) = $year";
    $lastDate = $db->createCommand($queryLast)->queryOne();

    $date = date('d F Y', strtotime($lastDate['day']));

    $queryCoa   =   "SELECT   gl_group.coa_name   as   name,
     gl_group.id from gl_group ORDER BY id ASC";
    $coaList = $db->createCommand($queryCoa)->queryAll();
    $models = $this->actionCari(0, $year, 1, 0);
    foreach ($models as $key => $value) {
        if($value['level'] == 1 && $mark != 1) {
            $profit += $value['balance'];
            $lastProfit += $value['balanceOld'];
            $mark++;
        }

        else if($value['level'] == 1 && $mark == 1) {
            $profit -= $value['balance'];
            $lastProfit -= $value['balanceOld'];
            $mark++;
        }
    }
    return $this->render('yearly', [
         'models' => $models,
         'dataYear' => $dataYear,
         'year' => $year,
         'profit' => $profit,
         'lastProfit' => $lastProfit,
         'coaList' => $coaList,
         'date' => $date,
    ]);
}
```

Kode Sumber B.9. Kode Sumber Melihat Laporan *Income Statement*

```
public function actionYearly($year = NULL){

    $queryGet = "SELECT gl_journal_h.id from gl_journal_h,
     gl_period   where   gl_journal_h.Post   =   1   and
     YEAR(gl_period.period_name) = $year and gl_period.id =
     gl_journal_h.period_id";
    $getJournal      =      $db->createCommand($queryGet)-
     >queryAll();
    foreach ($getJournal as $key => $value) {
        $queryLine = "SELECT gl_account.acc_name as name,
     gl_account.coa_id as coa, gl_journal_l.line_debit as
     debit,   gl_journal_l.line_credit   as   credit   from
     gl_account, gl_journal_l where gl_journal_l.journal_id
     = $value[id] and gl_journal_l.acc_id = gl_account.id";
        $getLineAcc    =    $db->createCommand($queryLine)-
     >queryAll();
        $lines[$key] = $getLineAcc;
    }
    foreach ($lines as $key => $value) {
        foreach ($value as $key1 => $value1) {
            $output[$count] = $value1;
            $count++;
        }
    }
    foreach ($output as $key => $value) {
        for($i = $key+1; $i <= $count;$i++) {
            if(!isset($output[$i])) continue;
            if($i >= $count) break;
            if($value['name'] == $output[$i]['name']) {
                $output[$key]['debit']                    +=
     $output[$i]['debit'];
                $output[$key]['credit']                   +=
     $output[$i]['credit'];
                unset($output[$i]);
            }
        }
        if(isset($output[$key])) {
            $output[$key]['balance']                       =
     $output[$key]['credit'] - $output[$key]['debit'];
            if($output[$key]['balance'] == 0) {
                unset($output[$key]);
            }
        }
    }
    return $this->render('yearly', [
        'acc' => $acc,
        'output' => $output,
        'year' => $year,
```

```
            'date' => $date,
        ]);
}
```

Kode Sumber B.10. Kode Sumber Melihat Laporan *Cash Flow*

Pada Kode Sumber B.8, ditunjukkan potongan kode sumber untuk menampilkan laporan *balance sheet.* actionYearly() adalah fungsi untuk menampilkan laporan *balance sheet* untuk periode tahunan.

Pada Kode Sumber B.9, ditunjukkan potongan kode sumber untuk menampilkan laporan *income statement.* actionYearly() adalah fungsi untuk menampilkan laporan *income statement* untuk periode tahunan.

Pada Kode Sumber B.10, ditunjukkan potongan kode sumber untuk menampilkan laporan *cash flow.* actionYearly() adalah fungsi untuk menampilkan laporan *cash flow* untuk periode tahunan.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan tugas akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

6.1 Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Pada tugas akhir ini berhasil dihasilkan aplikasi berupa ERP modul *general ledger* yang dapat menggantikan proses akuntansi manual karena dapat membantu para akuntan dalam proses penjurnalan serta penelusuran jurnal secara mudah. Selain itu proses pelaporan hasil keuangan juga dapat dibuat dengan mudah, karena sudah terkoneksi dengan jurnal-jurnal yang telah masuk ke dalam sistem.
2. Algoritma *tree* berhasil diimplementasikan untuk pembuatan *chart of accounts*, sehingga dapat meminimalisasi kerja yang harus dilakukan untuk membuat suatu akun finansial.
3. *Multitenancy* berhasil diimplementasi sehingga pengguna dapat mengakses aplikasi yang sama akan tetapi dengan data dari masing-masing *tenant*. Dalam pengaplikasian *multitenancy* dibantu dengan adanya *Role Based Access Control (RBAC)* untuk dapat membedakan kepemilikian data.
4. Basis data terdistribusi digunakan sebagai solusi untuk dapat menyelesaikan masalah ketika salah satu *tenant* mengalami *fail-over* dan *tenant* lain tetap dapat mengakses aplikasi, dengan menggunakan replikasi dan fragmentasi yang disediakan oleh MySQL Cluster.

6.2 Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan.

1. Pengembangan terkait cara pembuatan akun dengan algoritma lainnya. Lalu bandingkan dengan algoritma yang sudah ada pada aplikasi yang telah dirancang.
2. Adakan sistem yang berorientasi *mobile* pada modul *general ledger*. Sehingga dapat meningkatkan mobilitas dari para pengguna dan dapat memudahkan apabila pengguna merupakan pekerja dengan mobilitas yang cukup tinggi.

DAFTAR PUSTAKA

[1] S. R. Magal and J. Word, Integrated Business Process with ERP System, Danvers: John Wiley & Sons, Inc., 2012.

[2] N. J. Beauchamp, "Basics of Finance and Accounting," *Journal of the American College of Radiology,* vol. 11, no. 6, pp. 541-542, 2014.

[3] Hery, Pengantar Akuntansi, Jakarta: Grasindo, 2015.

[4] Ikatan Akuntan Indonesia, Standar Akuntansi Keuangan, Jakarta: Salemba Empat, 2009.

[5] R. Sarno, D. Sunaryono, V. Hariadi and Y. Kurniawan, "Perancangan dan Pembangunan Perangkat Lunak Berorientasi Arsitektur Servis (SOA) Dengan Pendekatan Workflow Pada Domain Cash Bank dan General Ledger ERP," *SESINDO 2013,* 2013.

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Muhammad Ashari Adhitama, lahir di Jakarta, 21 Juni 1994. Penulis menempuh pendidikan dasar mulai kelas 1 sampai 6 di SD Islam Al-Azhar 6 Jakapermai, Bekasi. Untuk pendidikan menengah, penulis menempuhnya di SMP Islam Al-Azhar 6 Jakapermai, Bekasi dan dilanjutkan di SMA Islam Al-Azhar 1 Pusat, Jakarta. Penulis melanjutkan pendidikan sarjana di Jurusan Teknik Informatika, Institut Teknologi Sepuluh Nopember Surabaya.

Penulis dalam menyelesaikan pendidikan S1 mengambil bidang minat Manajemen Informasi (*Information Management*) dan memiliki ketertarikan di bidang *Web Application Development*, serta *Financial Accounting*. Penulis dapat dihubungi melalui email : ashariad21@gmail.com.